Tampereen yliopisto

Juho Kuusisto

SUURTEN DATAMÄÄRIEN INTERAKTIIVINEN VISUALISOINTI SELAINYMPÄRISTÖSSÄ

Informaatioteknologian ja viestinnän tiedekunta
Diplomityö
Heinäkuu 2019

TIIVISTELMÄ

Juho Kuusisto: Suurten datamäärien interaktiivinen visualisointi selainympäristössä
Diplomityö
Tampereen yliopisto
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Heinäkuu 2019

Nykyaikaiset verkkoselaimet tarjoavat kattavat mahdollisuudet grafiikan esittämiseen Canvas- ja WebGL-ohjelmointirajapintojen kautta. Tässä diplomityössä kehitettiin grafiikkasuoritinta hyödyntävä pistekaaviovisualisointi React-komponenttina WebGL-ohjelmointirajapinnan tarjoamia ominaisuuksia käyttäen. Tämä diplomityö toteutettiin suomalaisen maailmanlaajuisesti toimivan tietoliikennealan yhtiön toimeksiannosta.

Tässä diplomityössä perehdytään pintapuolisesti datan visualisoinnissa käytettyihin kaaviotyyppeihin, interaktiivisen visualisoinnin ominaispiirteisiin ja datan visualisointiin selainympäristössä. Pääasiallinen huomio on kehitettävän visualisointikomponentin suorituskykymittausmenetelmien laatimisessa, visualisointikomponentin toteutuksessa ja visualisointikomponentin suorituskyvyn mittaamisessa.

Toteutettu visualisointikomponentti kykenee esittämään nykyaikaisilla laitteilla ongelmitta ja suorituskyvyn heikkenemättä 3 000 000 datapistettä ja 10 vuotta vanhalla kannettavalla tietokoneella 1 000 000 datapistettä hieman heikentyneellä suorituskyvyllä. Suorituskyky saavutti ja ylitti asetetut ennakko-odotukset. Visualisoinnin toteuttamisessa käytettiin Three.js-kirjastoa, joka tarjoaa ohjelmistokehittäjäystävällisen rajapinnan grafiikan luomiseen ja esittämiseen WebGL-ohjelmointirajapintaa hyödyntäen.

Visualisointikomponentin lisäksi tämän diplomityön sivutuotteena syntyi testipenkki React-applikaatio, jota käytettiin visualisointikomponentin suorituskyvyn testaamiseen. Testipenkki soveltuu myös muiden komponenttien kuin työssä kehitetyn testaamiseen.

Visualisointikomponenttia tulee testata tässä diplomityössä käytettyjä laitteita laajemmalla laitekannalla ennen laajamittaista hyödyntämistä. WebGL on hyvin tuettu, mutta kuten tässä työssä saadut tulokset osoittavat, suorituskyky vaihtelee laitekohtaisesti merkittävästi.

Avainsanat: visualisointi, WebGL, React

ABSTRACT

Juho Kuusisto: Interactive visualization of large amounts of data in browser environment
Master of Science Thesis
Tampere University
Master's Degree Programme in Information Technology
July 2019

Modern web browsers provide comprehensive possibilities for drawing graphics via Canvas and WebGL application programming interfaces. The goal for this thesis was to develop a scatter plot visualization as a React component that utilizes the features provided by WebGL application programming interface. The thesis was commissioned by a Finnish company operating in the telecommunication industry worldwide.

The subjects studied in this Master of Science Thesis are chart types used for data visualization, the characteristics of interactive visualization and data visualization in web browser environment. However, the focus is on developing the method for performance testing of the visualization component, developing the visualization component and measuring the performance of the visualization component.

The implemented visualization component can present 3 000 000 datapoints on modern devices without performance hit and 1 000 000 datapoints on a ten-year-old laptop with small performance hit. Performance reached and exceeded the set expectations. Three.js library was used for implementing the visualization. Three.js provides a software developer friendly application programming interface for creating and displaying graphics using WebGL.

In addition to the visualization component, a testbench React application was created as a by-product. The testbench was used for performance testing the visualization component.

The visualization component should be tested with wider range of devices than the ones used in this thesis before any extensive use. WebGL is well supported, but as the results of the performance tests show, the performance varies from device to device.

Keywords: visualization, WebGL, React

ALKUSANAT

Haluan kiittää työni ohjaajaa ja työkaveriani Jussia tarvittavan ohjauksen antamisesta. Lisäksi haluan kiittää vanhempiani Jaanaa ja Aria, ystävääni Valtteria ja pomoani Ninaa työn luettavuutta koskeneista huomioista. Kiitos myös kaikille työn etenemisestä kyselleille ja täten työn tekemiseen patistaneille tahoille.

Tampereella, 10.7.2019

Juho Kuusisto

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

API	engl. Application Programming Interface, ohjelmointirajapinta
CSS	engl. Cascading Style Sheets, HTLM-dokumenttien tyyliohjeet
DOM	engl. Document Object Model, dokumenttioliomalli
DPIP	Datapistemäärä pienessä päivityksessä
DPTP	Datapistemäärä täydessä päivityksessä
fps	engl. Frames Per Second, kehyksiä sekunnissa
GLSL ES	engl. OpenGL ES Shading Language, varjostinkieli
HTML	engl. Hypertext Markup Language, hypertekstin merkintäkieli
OpenGL ES	engl. Open Graphics Library for Embedded Systems, avoin grafiikkakirjasto sulautetuille järjestelmille
props	React-komponentin parametreista käytetty nimitys
SMIL	engl. Synchronized Multimedia Integration Language, XML-pohjainen kuvauskieli multimediaesityksille
SPA	engl. Single-page-application, selainpohjainen sovellus, joka dynaamisesti muuttaa verkkosivun sisältöä
SVG	engl. Scalable Vector Graphics, skaalautuva vektorigrafiikka
WebGL	engl. Web Graphics Library, verkkografiikkakirjasto
XML	engl. Extensible Markup Language, merkintäkieli

1. JOHDANTO

Mobiiliverkkojen seurantaan käytettävät järjestelmät tuottavat huomattavan määrän dataa, jonka esittäminen käyttäjälle on haastavaa. Yksi tapa esittää tätä kerättyä dataa on maantieteellisen kontekstin hyödyntäminen. Tämä voidaan toteuttaa esimerkiksi pistekaaviona karttapohjan päällä. Tällöin käyttäjän on helppoa verrata lähekkäin olevien tukiasemien tilaa toisiinsa ja tunnistaa vikatilanteiden laajuus.

Nykyaikaiset verkkoselaimet tarjoavat Canvas- ja WebGL-rajapintojen kautta mahdollisuuden edistyneiden graafisten ominaisuuksien luomiseen. Canvas-rajapinnan kautta ohjelmistokehittäjä pystyy piirtämään JavaScriptillä kaksiulotteisia muotoja ja bittikarttakuvia [1]. WebGL (engl. Web Graphics Library, suomennettuna verkkografiikkakirjasto) on huomattavasti matalamman tason rajapinta, tarjoten mahdollisuuden suorittaa GLSL ES -varjostinkielellä kirjoitettuja varjostinohjelmia laitteen grafiikkasuorittimella. Näin ollen WebGL-rajapinnan kautta on mahdollista hyödyntää grafiikkasuorittimen tarjoamaa laskentatehoa osana selainpohjaista sovellusta, esimerkiksi monimutkaisten ja laajojen visualisointien esittämiseen.

Tässä diplomityössä perehdytään suurten datamäärien interaktiiviseen visualisointiin selainympäristössä hyödyntäen WebGL-rajapintaa. Tavoitteena on kehittää prototyyppi pistekaaviovisualisoinnista React-komponenttina, joka on mahdollista ottaa osaksi olemassa olevaa järjestelmää. Prototyypin suorituskyky testataan monipuolisesti useammalla laitteella.

Tutkimusmenetelmänä työssä käytetään konstruktiivista tutkimusta, jossa pyritään ratkaisemaan todellisia ongelmia luomalla uusia konstruktioita. Konstruktio on abstrakti käsite, jolle on rajaton määrä mahdollisia toteutumia. Niille on tunnusomaista se, että ne eivät ole löydettyjä, vaan ne keksitään ja kehitetään. Konstruktiivisessa tutkimuksessa on oleellista toteuttaa kehitetty konstruktio, jolloin sen käytäntöön soveltuvuutta testataan. [2]

Kehitettävää prototyyppi ja sen testaamista varten kehitettävä testipenkki ovat konstruktioita. Prototyypin käytännön soveltuvuutta testataan testipenkin avulla. Vastaavasti testipenkin soveltuvuus tulee todistettua prototyyppiä testattaessa.

Diplomityö voidaan jakaa kahteen osaan: taustaosa ja toteutusosa. Taustaosa koostuu luvuista 2-4. Luvussa 2 esitellään datan visualisoinnin perusteita ja esitellään yleisimmin

käytetyt kaaviotyypit. Luvussa 3 perehdytään visualisointien toteuttamiseen selainympäristössä. Luvussa 4 esitellään prototyypin testaamiseen käytettävä menetelmä ja testitapaukset.

Toteutusosa koostuu luvuista 5-7. Luvussa 5 käydään läpi prototyypille asetetut vaatimukset ja esitellään toteutetun prototyypin toimintaperiaate. Luvussa 6 esitellään suorituskykytesteistä saadut tulokset. Lopuksi luvussa 7 on diplomityön johtopäätökset ja jatkokehitysideat.

2. DATAN VISUALISOINTI

Tässä luvussa perehdytään datan visualisointiin, yleisesti käytettyihin kaaviotyyppeihin ja interaktiivisen visualisoinnin tuomiin etuihin.

Saatavilla olevan datan määrä on muuttunut Internetin myötä merkittävästi, eivätkä Internet of Things -laitteet, sosiaalinen media ja käyttäjien seurannan kehittyminen ainakaan hidasta kerätyn datan volyymia. Tämän kehityksen myötä datan visualisoinnin kehittyminen on yhä tärkeämpää. Visualisoinnin avulla ihmiset kykenevät tutkimaan ja ymmärtämään kerättyä dataa. Waren [3] mukaan visuaaliset kanavat tarjoavat eniten tiedonsiirtokapasiteettia ihmisen ja tietokoneen välillä.

Visuaalianalytiikka (engl. Visual Analytics) on iteratiivinen prosessi, joka pyrkii ymmärtämään käsiteltävää ongelmaa saatavilla olevan datan visuaalisen esityksen kautta. Visuaalianalytiikka on visualisointia laajempi käsite, joka kattaa datan käsittelyä ja ymmärtämistä sen esittämisen lisäksi. [4]

2.1 Tyypillisiä visualisointiin käytettyjä kaaviotyyppejä

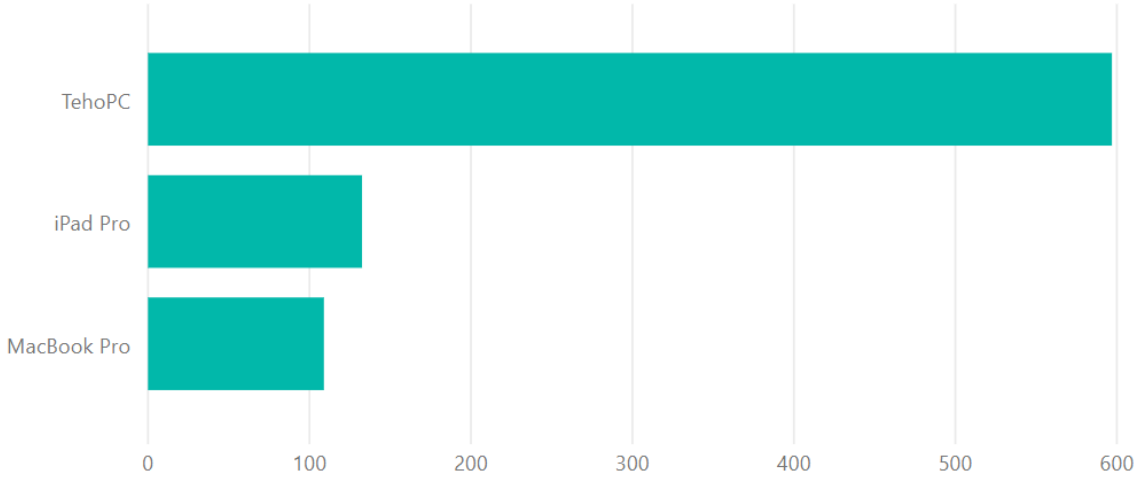
Perinteisiä visualisointiin käytettyjä kaaviotyyppejä ovat muun muassa pylväs-, linja-, piste- ja piirakkakaavio. Näistä kullakin on vahvuutensa ja ne sopivat erilaisten tietojen esittämiseen. Aliluvuissa 2.2.1-2.2.4 perehdytään jokaiseen yksitellen käyttäen lähteenä Vesa Kuuselan Tilastografiikan perusteet -oppikirjaa [5] ja esimerkkeinä tämän diplomityön suorituskykymittausten mittausdatasta tehtyjä kaavioita.

Mainituista kaavioista William Playfairin (1759-1823) ideoimia, yleisen mielipiteen mukaan, ovat pylväskaavio, linjakaavio ja piirakkakaavio. Nykyaikainen pistekaavio on sen sijaan kuvattu ensimmäisen kerran J. F. W. Herschelin 1833 julkaisemassa artikkelissa, joskin pistekaavion juuret ovat sitä vanhemmat. [6]

2.1.1 Pylväskaavio

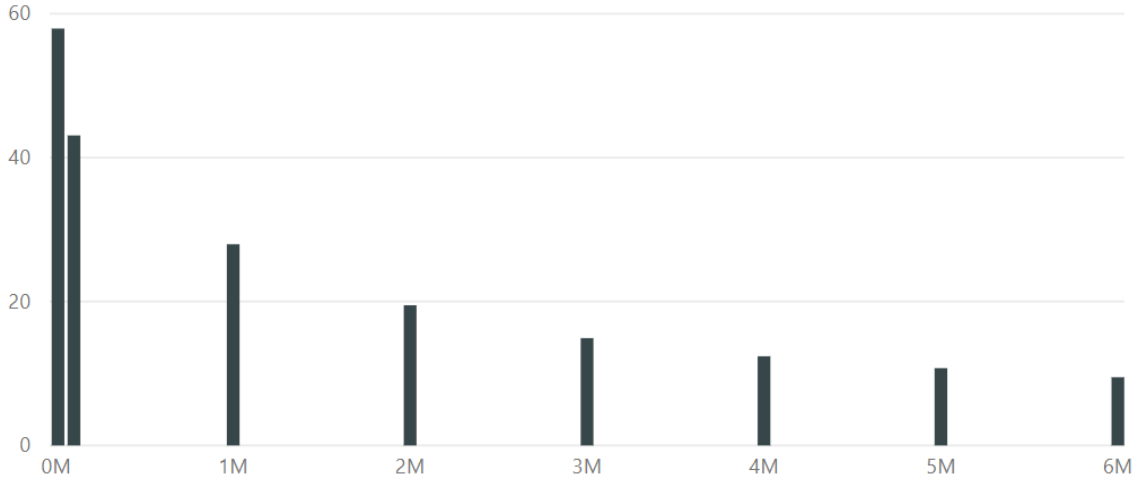
Pylväskaaviolla tarkoitetaan kuviota, jossa joko vaaka- tai pystyakselille on sijoitettu yleensä samanlevyisiä suorakaiteita. Pylväskaaviot soveltuvat ilmaisemaan määrää. Pysty- ja vaakapylväskaaviot eivät ole keskenään vaihtoehtoisia tilanteessa kuin tilanteessa, vaan kummallakin on omanlaisensa käyttökohteet. Pystypylväskaavio soveltuu käytettäväksi, kun x-akselilla on jatkuva-arvoinen ominaisuus. [5]

Kuvassa 1 on esimerkki vaakapylväskaaviosta. Esimerkin kaavio kertoo diplomityössä käytettyjen testilaitteiden millisekunnissa generoimien datapisteiden lukumäärän. Tähän tarkoitukseen pystypylväskaavio ei sovellu, sillä testilaite ei ole jatkuva-arvoinen ominaisuus.



Kuva 1. *Esimerkki vaakapylväskaaviosta, diplomityössä käytettyjen testilaitteiden millisekunnissa generoimien datapisteiden lukumäärä.*

Kuvassa 2 on esimerkki pystypylväskaaviosta, joka kertoo keskimääräisen ruudunpäivitysnopeuden kehyksinä sekunnissa MacBook Prolle. X-akselilla on datapistemäärä, jota voidaan pitää jatkuva-arvoisena ominaisuutena. Käytetyn asteikon tulee olla tasavälinen, jotta kaavio ei vääristä esitettyä informaatiota [5].



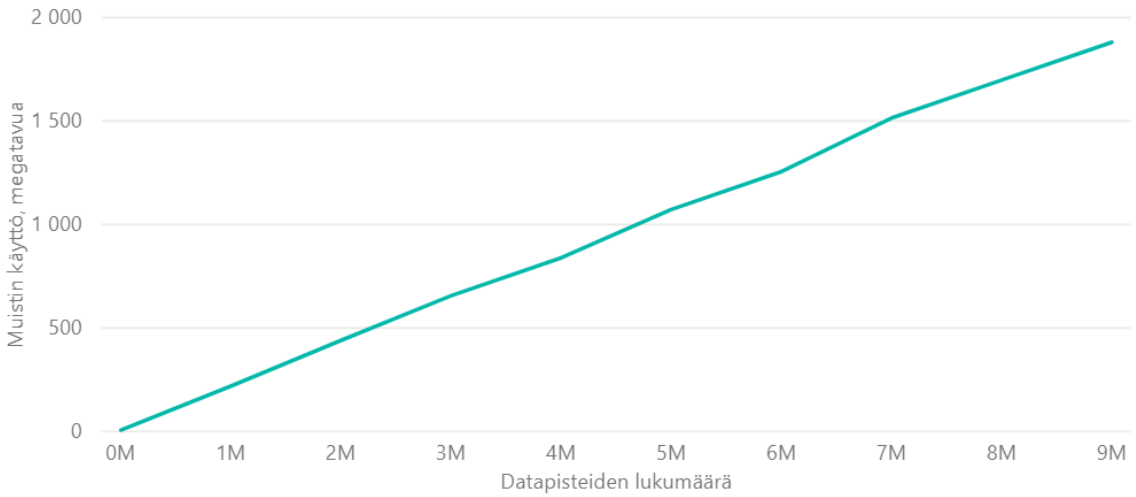
Kuva 2. *Esimerkki pystypylväskaaviosta, diplomityössä toteutetun visualisoinnin keskimääräinen ruudunpäivitysnopeus datapistemäärän funktiona MacBook Prolle.*

Vaakapylväskaaviolle usein vaihtoehtoinen kaaviotyyppi on linjakaavio, jolle pätee yhtäläinen rajoitus x-akselin ominaisuuden jatkuva-arvoiselle luonteelle. [5]

2.1.2 Linjakaavio

Linjakaavio, joka tunnetaan myös nimellä viivakuvio tarkoittaa kaaviota, jossa koordinaatistoon merkittyjen pisteiden kautta on vedetty viiva tai viivoja. Tyypillistä on, että toisena koordinaattina on määrää kuvaava lukuarvo ja toisena selittävä tekijä, usein aika. Oleellista linjakaaviossa on pisteiden järjestys ja oletus niiden välisen muutoksen luonteesta. Linjakaaviolla kuvataan ensisijaisesti muutosta tai kehitystä, mistä johtuu linjakaavion vaatimus siitä, että kummallakin akselilla on oltava jatkuva-arvoinen ilmiö. [5]

Kuvassa 3 on esimerkki linjakaaviosta. Y-akselilla on esitetty visualisoinnin muistin käyttö megatavuina ja x-akselilla datapisteiden lukumäärä. Kummatkin ovat jatkuva-arvoisia ominaisuuksia. Akseleiden valinnassa on noudatettu Vesa Kuuselan ohjeistusta ”x-akselille [on] pantava se ominaisuus, joka on syy, ja y-akselille seuraus” [5].



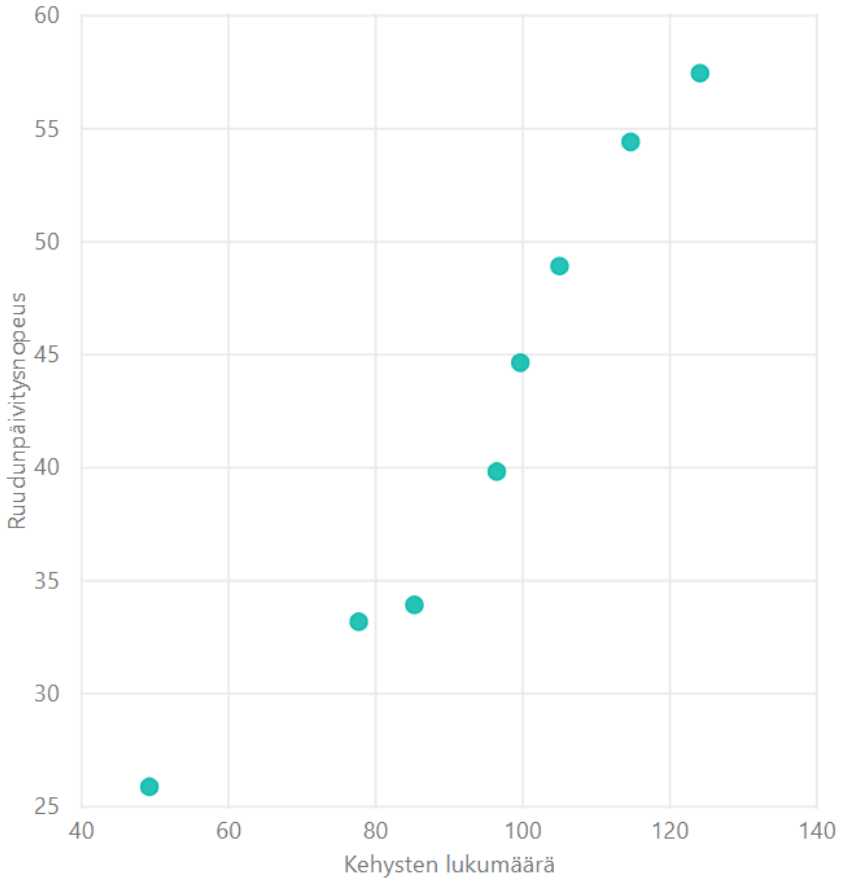
Kuva 3. *Esimerkki linjakaaviosta, diplomityössä kehitetyn visualisointikomponentin muistinkäyttö datapisteiden lukumäärän suhteen.*

2.1.3 Pistekaavio

Pistekaaviossa havainnot piirretään suorakulmaiseen x-y-koordinaatistoon. Pistekaavion avulla muuttujien väliset riippuvuudet on helppo havaita ja arvioida niiden voimakkuutta. Suoraviivaisen riippuvuuden lisäksi pistekaaviosta on mahdollista havaita myös erikoisemmat riippuvuudet, jos pisteet muodostavat esimerkiksi paraabelin muotoisen parven. Riippuvuudet voidaan toki osoittaa myös laskennallisin keinoin, mutta pistekaavion avulla tehtävä helpottuu. [5]

Kuvassa 4 on esimerkki pistekaaviosta. Y-akselilla on ruudunpäivitysnopeus kehyksinä sekunnissa ja X-akselilla kehysten lukumäärä testin aikana. Esimerkki on toki melko

epäolennainen, mutta osoittaa oivallisesti näiden muuttujien välisen suoran riippuvuuden.



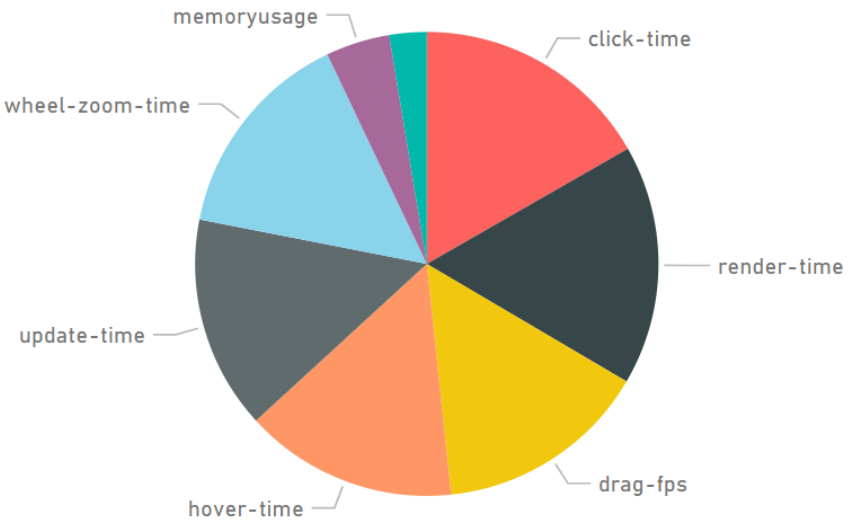
Kuva 4. *Esimerkki pistekaaviosta, y-akselilla ruudunpäivitysnopeus kehyksinä sekunnissa ja x-akselilla kehysten lukumäärä.*

Diplomityössä kehitettävä visualisointikomponentti on tyypiltään pistekaavio.

2.1.4 Piirakkakaavio

Piirakkakaavio esittää vertailtavien luokkien suhteellisia osuuksia luokkien esittämästä kokonaisuudesta. Esimerkiksi laajalevikkisissä julkaisuissa piirakkakaavio on yleisimmin käytetty kaaviotyyppi siksi, että sen pyöreä muoto koetaan kulmikkaita esityksiä vetoavampana. Piirakkakaavion heikko visuaalinen vertailtavuus ja alhainen tarkkuus ovat herättäneet arvostelua. Usein erityisesti tarkkuutta vaativissa tehtävissä jokin toinen kaavio tyyppi onkin onnistuneempi valinta. [5]

Kuvassa 5 on esitelty erilaisten testiparametriyhdistelmien määrän jakauma testitapauksittain piirakkakaaviona. Esimerkistä on helppo havaita mainittuja piirakkakaavion ongelmia; kuusi suurinta luokkaa ovat lähes saman suuruisia, jolloin niiden välisten erojen hahmottaminen kaaviosta on mahdotonta.



Kuva 5. *Esimerkki piirakkakaaviosta, erilaisten testiparametrikonfiguraatioiden jakauma testitapauksittain. Pienimmän viipaleen otsikko continous-data ei mahtunut kuvaan.*

2.2 Interaktiivinen visualisointi

Usein data on luonteeltaan sellaista, että se voidaan visualisoida usealla eri tavalla, eikä ole selvää mikä on paras tapa [4]. Osaratkaisu ongelmaan on pyrkiä kehittämään dataan sopiva visualisointi hyödyntäen perinteisistä kaaviotyypeistä poikkeavia tapoja ja tarjota käyttäjälle mahdollisuus olla vuorovaikutuksessa visualisoinnin kanssa. Voidaan myös sanoa, että jopa staattisten visualisointien, kuten julisteiden, kanssa käyttäjät interaktioivat esimerkiksi kääntämällä julistetta ja tutkimalla sitä sekä lähempää että kauempaa [7].

J. S. Yi et al. määrittelevät seitsemän kategoriaa interaktioille tiedon visualisoinnissa [7]. Nämä kategoriat on listattu taulukossa 1.

Taulukko 1 *Seitsemän interaktiokategoriaa ja kuvaukset J. S. Yi et al. mukaan [7]*

Kategoria	Kuvaus
Valitse (select)	Merkitse jokin mielenkiintoiseksi (esimerkiksi paikkamerkinnät Google Mapsissa)
Tutki (explore)	Näytä jotakin muuta (esimerkiksi ison kaavion panorointi)
Uudelleenmäärittele (reconfigure)	Näytä eri järjestys (esimerkiksi järjestä taulukkomuotoinen visualisointi kolumnin mukaan)
Muunna (encode)	Näytä eri esitysmuoto (esimerkiksi muunna piirakkakaavio histogrammiksi)
Abstrahoi/selitä (abstract/elaborate)	Näytä enemmän tai vähemmän yksityiskohtia (esimerkiksi vihjetekstin näyttäminen tai zoomaus niin, että lähemmäksi mentäessä yksityiskohdat lisääntyvät)
Suodata (filter)	Näytä jotakin ehdollisesti (esimerkiksi näkyvien datapisteiden valinta jonkin käyttäjän määrittelemän ehdon mukaisesti, kuten esimerkiksi ajan)
Yhdistä (connect)	Näytä toisiinsa liittyvät esineet (esimerkiksi ajatuskartassa korosta suorat naapurit)

Diplomityössä kehitettävä visualisointi tukee yllä esitellyn luokittelun mukaisista interaktioista valitse-, tutki- ja abstraktoi/selitä-tyyppisiä interaktioita. Valitse-tyyppinen interaktio toteutetaan siten, että visualisoinnissa on mahdollista valita yksittäinen datapiste. Tutki-tyyppinen interaktio toteutetaan visualisoinnin panorointina hiirellä raahatessa ja Abstrahoi/selitä-tyyppinen interaktio visualisoinnin zoomauksena hiiren rullalla.

3. DATAN VISUALISOINTI SELAINYMPÄRISTÖSSÄ

Tässä luvussa perehdytään selainympäristössä käytettävissä oleviin datan visualisointiin soveltuviin rajapintoihin ja työkaluihin. Aliluvussa 3.1 perehdytään lyhyesti JavaScriptiin, joka on selainympäristössä suoritettava ohjelmointikieli. Aliluvuissa 3.2-3.4 käydään läpi yleisimmin interaktiivisten visualisointien rakentamiseen käytetyt rajapinnat. Aliluvussa 3.5 esitellään näiden rajapintojen käyttöä helpottavia, yleisesti käytettyjä kirjastoja. Lopuksi aliluvussa 3.6 perehdytään selainpohjaisiin sovelluksiin yleisellä tasolla.

3.1 JavaScript

JavaScript on Brendan Eichin vuonna 1995 kehittämä ohjelmointikieli. Motiivina hänellä oli luoda tapa, jolla voitiin ohjelmoida aikansa suosituinta verkkoselainta, Netscape Navigatoria. Vuosien saatossa JavaScriptiä ovat alkaneet tukemaan muutkin verkkoselaimet ja toisistaan eroavat JavaScript-toteutukset ovat sulautuneet yhdeksi standardiksi. [8]

Merkittävä askel JavaScriptin suosion kasvuun on ollut Node.js:n tuoma mahdollisuus JavaScriptin ajamiseen selaimen lisäksi palvelimella. Niin sanottu ”JavaScript kaikkialla” -liike pyrkii vähentämään asioiden määrää, jotka ohjelmistokehittäjän tulee hallita työssään [9]. Node.js:n myötä kehitetyt työkalut, kuten Node Package Manager pakettien hallintaa varten ja Webpack ja Browserify JavaScript-tiedostojen paketointiin ja valmisteluun selaimessa ajoa varten, ovat yhä kiihdyttäneet JavaScriptin suosiota ohjelmistokehittäjien keskuudessa.

JavaScriptiä ei tule sekoittaa Java-kieleen, jonka kanssa sillä ei ole mitään tekemistä. Voitaneen sanoa, että JavaScriptin syntaksi perustuu löyhästi C-ohjelmointikieleen. JavaScript on dynaamisesti tyypitetty, automaattisella roskienkeruulla varustettu, tulkattava kieli.

3.2 SVG

SVG (engl. Scalable Vector Graphics, suomennettuna skaalautuva vektorigrafiikka) on XML-pohjainen (engl. Extensible Markup Language, suomennettuna laajennettava merkintäkieli) teknologia, joka tukee 2D-muotoja. SVG mahdollistaa yksinkertaisten viivojen lisäksi erilaisten gradienttien, efektien ja muunnosten määrittelyn. SVG on myös kuvatiedostoformaatti. [10]

SVG on eksplisiittisesti suunniteltu toimimaan muiden web-standardien, kuten CSS:n (engl. Cascading Style Sheets, suomennettuna porrastetut tyyliarkit), DOMin (engl. Document Object Model, suomennettuna dokumenttioliomalli) ja SMILin (engl. Synchronized Multimedia Integration Language, XML-pohjainen kuvauskieli multimediaesityksille), kanssa. SVG on World Wide Web Consortiumin 1999 kehittämä avoin standardi. [11]

SVG eroaa Canvas APIsta ja WebGL:stä siinä, että SVG on osa DOMia [12], kun taasen Canvas API ja WebGL piirtävät HTML5 canvas-elementin sisälle irrallaan DOM:stä. Tämän ansiosta SVG:llä toteutetun visualisoinnin osiin on mahdollista liittää tapahtumakuuntelijoita samaan tapaan kuin muihinkin sivulla esiintyviin elementteihin.

3.3 Canvas API

Canvas API mahdollistaa grafiikan piirtämisen JavaScriptillä HTML5 canvas-elementtiin [1]. Canvas API käyttää piirtoon niin kutsuttua *immediate mode*a, eli canvas-elementti ei tiedä piirto-operaation jälkeen mitään juuri piirretystä muodosta. Ohjelmistokehittäjä on vastuussa kohteiden piirtämisestä, lopputuloksesta ja kuvan päivittämisestä [13]. Hyvä vertauskuva onkin aito paperi; piirtämisestä jää jäljelle vain väripigmenttejä.

Canvas API on pikseleiden piirtämistä varten. Se ei tarjoa valmiita (skaalautuvia) muotoja tai vektoreita eikä elementtejä, joihin kehittäjä voisi kiinnittää tapahtumakäsittelijöitä. [14]

Canvas API on SVG:tä suorituskykyisempi rajapinta. Canvas APIa käytetään muun muassa animaatioihin, peligrafiikkaan, datan visualisointiin, valokuvien muokkaamiseen ja reaaliaikaiseen videoprosessointiin [1].

3.4 WebGL

WebGL on JavaScript-ohjelmointirajapinta interaktiivisen laitteistolla kiihdytetyn 3D- ja 2D-grafiikan piirtämiseen ilman selainlaajennoksia. WebGL on samankaltainen kuin OpenGL ES 2.0 (engl. Open Graphics Library for Embedded Systems, suomennettuna avoin grafiikkakirjasto sulautetuille järjestelmille) ja sitä voidaan käyttää HTML5 canvas -elementteihin piirtämiseen. Selaimista WebGL:ä tukevat viimeisimmät versiot Firefoxista, Google Chromesta, Operasta, Safarista, Internet Explorerista ja Microsoft Edgestä. Selaintuen lisäksi käytettävän päätelaitteen on tuettava ominaisuuksia laitteistotasolla. [15]

Khronos Group julkaisi vuonna 2011 WebGL-rajapinnan version version 1.0 [16]. Mukana kehitystyössä ovat olleet kaikki suuret selaintoimittajat: Apple, Google, Mozilla

ja Microsoft. WebGL-rajapinnan version 2 määrittely alkoi vuonna 2013. WebGL 2 -rajapinnan määrittelyn työluonnos on jo julkisesti saatavilla ja jotkin selaimet tukevat sitä.

Työn kirjoitushetkellä WebGL 1 -tuki alkaa löytymään lähes jokaisesta laitteesta ja WebGL 2 -tuki noin puolesta laitteista. [17]

3.5 Visualisointikirjastot

Natiivien rajapintojen käyttöä helpottamaan ja niiden ominaisuuksia täydentämään on saatavilla useita kolmannen osapuolen avoimen lähdekoodin JavaScript-kirjastoja. Siinä missä natiivit rajapinnat pyrkivät tarjoamaan mahdollisuuden vähän kaikkeen, keskittyvät kirjastot yleensä johonkin rajatumpaan käyttötapaukseen. Aliluvuissa 3.5.1-3.5.3 esitellään muutama yleisesti käytetty visualisointikirjasto.

3.5.1 D3.js

D3.js (myöhemmin D3) on JavaScript-kirjasto dokumenttien manipulointiin datan perusteella. D3 käyttää HTML:ä, SVG:tä ja CSS:ä visualisointien luomiseen. D3 pyrkii keskittymään tehokkuuteen ja monikäyttöisyyteen. [18]

D3:n on kehittänyt Michael Bostock et al. ratkaisuna aiempien kirjastojen ongelmiin, joita olivat muun muassa ongelmat saatavuuden kanssa ja rajoitettu ilmaisukyky. [19]

D3 on yksi tunnetuimmista visualisointien luomiseen käytetyistä JavaScript-kirjastoista.

3.5.2 Vega

Vega on deklaratiivinen visualisointikieli interaktiivisten visualisointien luomista varten. Visualisointi määritellään JSON-formaatissa mukaan lukien datalähteet, ulkonäkö ja interaktiot. Varsinainen visualisointi luodaan käyttäen joko Canvas APIa tai SVG:tä. [20]

Deklaratiivisen visualisointikielen ansiosta visualisointeja on nopeampi kehittää. Lisäksi se mahdollistaa sekä visualisointien siirtämisen alustojen välillä että ohjelmointikielikohtaisen optimoinnin [21]. Vega on aktiivisen tutkimustyön tulos, jonka kehittämisessä on hyödynnetty visualisointijärjestelmiin liittyvää tutkimusta yli vuosikymmenen ajalta [22].

3.5.3 Three.js

Three.js on JavaScript-kirjasto, joka helpottaa WebGL:n käyttöä tarjoamalla korkean tason JavaScript-ohjelmointirajapinnan 3D-grafiikan luomista varten. Three.js:n

ensimmäinen versio julkaistiin huhtikuussa 2010 [23]. Tämän jälkeen projekti on kerännyt yli 50 000 tähteä ja yli 1 000 kehittäjää GitHubissa [24].

Three.js tarjoaa tavan kuvata 3D-maailman puurakenteen avulla, melko vastaavasti kuin DOM kuvaa verkkosivun puurakenteena. Tästä puurakenteesta Three.js renderer-olio osaa piirtää näkymän verkkosivulle. [25]

3.6 Selainpohjaiset sovellukset

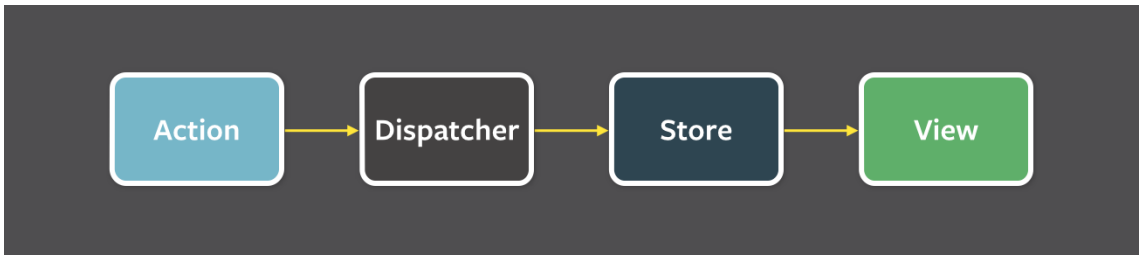
Nykyaikaiset selainpohjaiset sovellukset tunnetaan suomenkielessäkin lyhenteellä SPA, joka tulee englanninkielisistä sanoista single-page-application (suoraan suomennettuna yhden sivun sovellus). Ominaista näille sovelluksille on se, että verkkosivua ei ladata uudelleen avaamisen jälkeen, vaan sisältö muuttuu dynaamisesti ja tarvittavat resurssit noudetaan palvelimelta taustalla. Ohjelmointikielenä käytetään selainten tukemaa JavaScriptiä.

SPA-sovellusten kehittämiseen käytettyjä suosittuja kirjastoja ovat muun muassa React, Angular ja Vue.js. Tässä diplomityössä kehitettävä visualisointikomponentti toteutetaan React-komponenttina. Reactiin perehdytään tarkemmin aliluvussa 3.6.1.

3.6.1 React

Pete Hunt, Reactin ydintiimin jäsen, luonnehtii Reactia artikkelissaan ”Why did we build React?” (suomennettuna ”Miksi loimme Reactin?”) vapaasti suomentaen seuraavin sanoin: ”React on kirjasto koottavien käyttöliittymien rakentamista varten. Se kannustaa luomaan uudelleen käytettäviä käyttöliittymäkomponentteja, jotka esittävät ajan myötä muuttuvaa dataa” [26]. React ei siis ota kantaa kaikkiin perinteisen MVC-mallin mukaisiin kerroksiin, vaan keskittyy vain käyttöliittymään.

React-applikaatioiden tilan hallintaa varten on kehitetty useita ratkaisuja, joista yksi tunnetuimmista on Facebookin kehittämä flux-arkkitehtuuri. Keskeistä fluxissa on yksisuuntainen datavirta. Interaktiosta syntyy toimenpide (engl. action), joka välitetään keskitetyn lähettäjän kautta varastoon (engl. store), jonka perusteella käyttöliittymä päivittyy. Tätä havainnollistetaan kuvassa 6. [27]



Kuva 6. *Flux-arkkitehtuurin mukainen yksisuuntainen datavirta. Kuva fluxin dokumentaatiosta. [27]*

Flux-arkkitehtuurin pohjalta on luotu Redux-arkkitehtuuri ja -kirjasto. Keskeisimpiä eroavaisuuksia puhtaaseen flux-arkkitehtuuriin ovat yksi varasto usean sijaan, sovelluslogiikan erottaminen varastosta ja reducer-funktiot. Tässä diplomityössä kehitettävässä testipenkissä käytetään Redux-arkkitehtuuria.

4. VISUALISOINNIN SUORITUSKYVYN MITTAUSMENETELMÄ

Tässä luvussa esitellään visualisoinnin suorituskyvyn mittaamiseen käytettävä menetelmä. Suorituskyvyn mittauksessa keskitytään mitattavissa oleviin arvoihin. Huomiota ei kiinnitetä informaation välittymiseen käyttäjälle.

Mittauksen toteuttamiseksi ja visualisoinnin kehitystyön tueksi toteutetaan testipenkki-web-sovellus React-käyttöliittymäkirjastoa käyttäen ja Redux-arkkitehtuurin mukaisesti. Testipenkin pääasiallisia tehtäviä ovat:

- visualisointikomponentin esittäminen
- testidatan generointi
- testidatan syöttäminen visualisointikomponentille
- suureiden mittaaminen
- määriteltyjen testitapausten suorittaminen.

Testipenkkiä käyttämällä visualisoinnille voidaan ajaa toistettavia ja ennalta määriteltyjä testitapauksia useilla laitteilla ilman merkittävää manuaalista työtä.

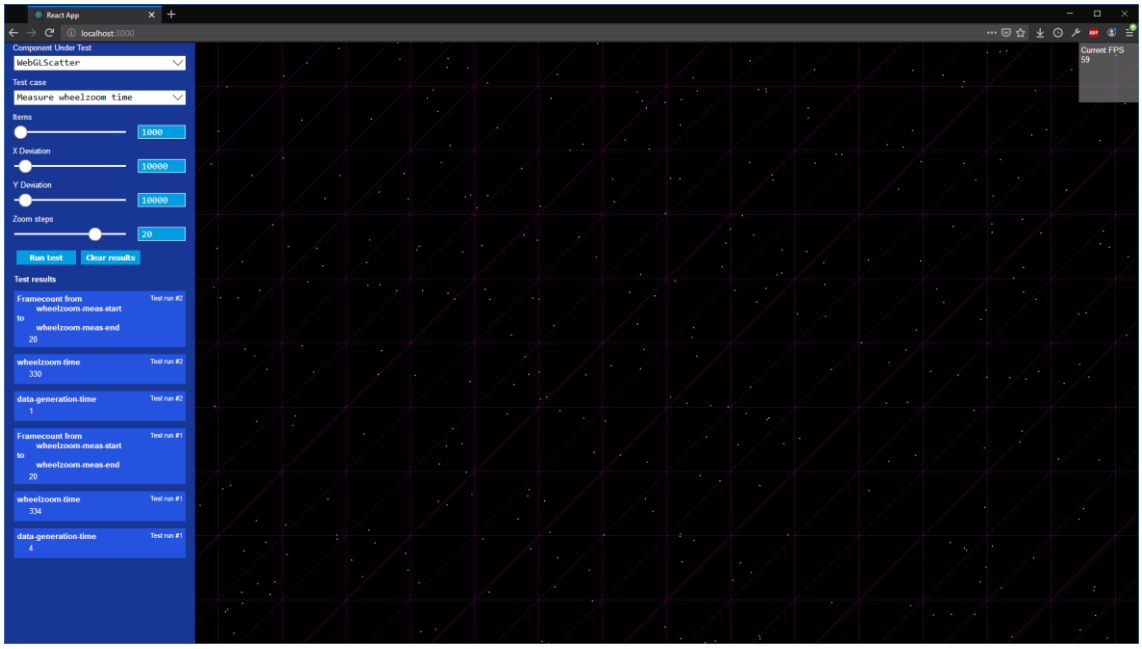
4.1 Testipenkin toteuttaminen

Testipenkki toteutetaan React-sovelluksena. Siinä missä uuden React-web-sovelluksen aloittaminen vaati ennen monimutkaisen kehitysympäristön pystyttämisen, onnistuu se nykyään Create React Appin [28] avulla. Create React App mahdollistaa uuden projektin aloittamisen yhdellä komennolla, jonka jälkeen ohjelmistokehittäjällä on valmis kehitysympäristö käytettävänään. Mikäli myöhemmin ilmenee tarve räätälöidä sovelluksen kehitysympäristöä, voi projektin viedä ulos (engl. eject) Create React Appista, jolloin ympäristön asetukset tulevat ohjelmistokehittäjän saataville. Näin toimittaessa myös kehitysympäristön moduulien ajan tasalla pitäminen jää ohjelmistokehittäjän harteille. Projektia ei ole mahdollista viedä takaisin Create React Appin hallinnoimaksi.

Testipenkin ja visualisointikomponentin kehitystyössä käytetään Create React Appia, eikä projektia viedä ulos.

4.1.1 Testipenkin käyttöliittymä

Vaikka testipenkki toteutettiin lähtökohtaisesti vain tämän diplomityön tarpeisiin, pyrittiin siitä luomaan helposti uudelleenkäytettävä. Testipenkin käyttöliittymä suunniteltiin erikseen ennen toteuttamista, joskin joitakin muutoksia suunnitelmaan tuli työtä tehdessä. Kuvassa 7 on esitelty testipenkin lopullinen käyttöliittymä



Kuva 7. *Testipenkin käyttöliittymä*

Käyttöliittymän vasemmassa reunassa on sivupalkki, josta löytyy testipenkin hallintaan tarvittavat asetukset. Näitä ovat:

- testattavan komponentin valinta
- testitapauksen valinta
- valittuun testitapaukseen liittyvät kontrollit
- testin käynnistäminen ja testitulosten pyyhkimisen painikkeet
- lista testituloksista.

Käyttöliittymäkuvassa on valittuna diplomityössä toteutettu visualisointikomponentti ja aliluvussa 4.5.4 esitelty testitapaus ”Visualisoinnin zoomaus hiiren rullalla”. Testipenkin käyttöliittymä ja testitapaukset on toteutettu englanniksi uudelleenkäyttömahdollisuuksien parantamiseksi.

Sivupalkin lisäksi testipenkki näyttää visualisoinnin ruudunpäivitysnopeuden oikeassa ylänurkassa. WebGL-visualisointien kanssa tämä on järkevä mittari, sillä visualisointia piirretään uudelleen koko ajan. SVG- ja Canvas API -pohjaisten visualisointien kanssa ruudunpäivitysnopeus ei välttämättä ole mielekäs mittari, mutta sillä ei ole testipenkin toteutuksen tai tämän diplomityön kannalta merkitystä.

Testipenkin yhdessä kehitysversiossa oli mukana myös ajossa olevan testiaskeleen ilmaiseva kenttä, mutta lopullisesta versiosta se poistettiin turhana. Pääosin testiaskeleet ovat niin nopeita, että testiaskeleen nimen käyttöliittymässä näyttämisestä aiheutuva suorituskykyhaitta on saatua informaatioarvoa suurempi.

Sivupalkki ei valitettavasti skaalaudu pienille näytöille oikein. Näin ollen pienimmät laitteet, jolla testipenkkiä voidaan tässä diplomityössä käyttää, ovat taulutietokoneet. Yhteensopivuuden parantamiseksi sivupalkki tulisi näyttää pienillä laitteilla koko ruudun kokoisena ja piilottaa testiajon ajaksi.

4.1.2 Testidatan generointi ja syöttäminen

Testidatan generointia varten testipenkkiin toteutettiin moduuli, joka mahdollistaa halutun testidatan generoinnin. Moduuli toteutettiin melko suoraviivaisesti niin, että testidatan formaatti on kovakoodattu. Generoitavan testidatan määrää tulee voida säätää käyttöliittymästä käsin.

Testidatan tulee vastata lopputuotteessa käytettävää dataa mahdollisimman tarkasti. Näin ollen jokaisen datapisteen tulee sisältää pisteen koordinaatit joko koordinaatteina tai valmiiksi visualisoinnin käyttämään koordinaatistoon muunnettuna, pisteen yksilöivä tunniste ja yksi tai useampia muu pisteeseen liittyvä data-arvo. Näiden vaatimusten pohjalta testidatana päädyttiin käyttämään ohjelman 1 mukaisia objekteja.

```
{
    "x": 0,
    "y": 1,
    "id": "yksiloiva-string-tyyppinen-id",
    "data-1": 1337,
    "data-2": "string-data"
}
```

Ohjelma 1. *Esimerkki generoitavan testidatan yksittäisestä datapisteestä*

Todellisen kaltaista dataa käytettäessä testien tuloksista voidaan arvioida visualisoinnin käytettävyyttä osana olemassa olevaa tuotetta. Tämä mahdollistaa diplomityössä toteutetun visualisoinnin jatkokehitystä koskevien päätösten tekemisen ilman ylimääräistä työn ulkopuolista vaivannäköä.

Generointimoduuli toteutettiin JavaScript-moduulina. Vaihtoehtoisten testidatalähteiden tai -generaattorien tuominen osaksi testipenkkiä onnistuu uusien JavaScript-moduulien avulla. Testipenkkiin ei toteutettu tässä vaiheessa erillistä rajapintaa vaihtoehtoisten moduulien yhtäaikaista käyttöä varten.

Testidatan generoimista erillisellä ohjelmalla ja verkon yli testipenkille syöttämistä pohdittiin, mutta sen tuomaa lisäarvoa ei lopulta nähty riittäväksi. Erillinen generaattori

olisi simuloinut paremmin todellista tilannetta, jossa data tulee järjestelmän ulkopuolelta. Testipenkin ensisijainen tehtävä on kuitenkin toimia testipenkkinä visualisoinnille, joten datan lähteellä ei sinällään ole merkitystä.

4.1.3 Testitapausten määrittely

Testien toistettavuuden ja helpon ylläpidettävyyden mahdollistamiseksi testipenkin testitapaukset määritellään JSON-tiedostojen avulla. Testitapaukselle määritellään nimi, käyttöliittymässä näkyvät parametrit, piilotetut parametrit ja testiaskeleet. Lyhyt esimerkki testitapaustiedoston ylimmän tason rakenteesta on esitelty alla olevassa ohjelmassa 2.

```
  {
      "name": "Measure render time",
      "id": "rendertime",
      "description": "This testcase measures the time that is spent on rendering the
visualization.",
      "testModule": null,
      "parameters": [],
      "steps": []
  }
```

Ohjelma 2. *Esimerkki testitapauksen ylimmän tason rakenteesta*

Ohjelmassa 2 esiintyvä testModule-parametri mahdollistaa viittaamisen kattavampia ominaisuuksia tarjoaviin testimoduuleihin. Tällaisia voisivat olla esimerkiksi moduulit, jotka integroivat testipenkin osaksi isompaa kokonaisuutta tai tarjoavat testattavana olevalle komponentille räätälöityjä testifunktioita. Tämän diplomityön käyttötapauksessa näille ei ole tarvetta.

Oleellisin osa testitapauksen määrittelyä ovat käytettävät parametrit (engl. parameters) ja askeleet (engl. steps). Parametreiksi kelpaa lista objekteja, joissa on vähintään vaaditut ominaisuudet asetettuna. Parametriobjektin tukemat ja vaatimat ominaisuudet on listattu taulukossa 2.

Taulukko 2 *Parametriobjektin tukemat ja vaatimat ominaisuudet*

Ominaisuus	Kuvaus	Vaadittu?
name	Parametrin nimi, voidaan esittää käyttöliittymässä. Tietotyyppi: String	Kyllä
id	Parametrin yksilöivä tunniste. Ei näytetä käyttöliittymässä. Tietotyyppi: String	Kyllä
type	Parametrin tyyppi, kelvollisia arvoja ovat *int* ja *hidden*. Tietotyyppi: String	Kyllä
default	Parametrin oletusarvo. Vain parametrille, jonka tyyppi on *int*. Tietotyyppi: Number	Kyllä, *int*-tyyppisille
value	Parametrin arvo. Vain parametrille, jonka tyyppi on *hidden*. Tietotyyppi: Mikä tahansa	Kyllä, *hidden*-tyyppisille
min	Parametrin minimiarvo, vain *int*-tyyppisille parametreille. Tietotyyppi: Number	Kyllä, *int*-tyyppisille
max	Parametrin maksimiarvo, vain *int*-tyyppisille parametreille. Tietotyyppi: Number	Kyllä, *int*-tyyppisille

Testipenkissä hidden-tyyppiset parametrit eivät ole käyttäjän muutettavissa käyttöliittymän kautta. Hidden-tyyppisiä parametreja voidaan käyttää testiaskelissa hyödyksi.

Testiaskeleet määrittelevät testin suorituksen yksityiskohtaisesti. Ne suoritetaan annetussa järjestyksessä, mutta testiaskeleessa kutsuttava funktio voi aiheuttaa asynkronisia sivuvaikutuksia. Testiaskellistaan voi antaa kahden tyyppisiä objekteja, testiaskeleita ja testiaskelryhmiä. Kukin testiaskel voi suorittaa enintään yhden testifunktion ja lähettää enintään yhden Redux-arkkitehtuurin toimenpiteen (engl. action). Testiaskelryhmä koostuu testiaskeleista ja ryhmää suoritetaan annetun intervallin välein, kunnes annettu suoritusaika täyttyy.

Testiaskeleen tukemat ja vaatimat ominaisuudet on esitelty taulukossa 3 ja testiaskel-ryhmän tukemat ja vaatimat ominaisuudet on esitelty taulukossa 4.

Taulukko 3 *Testiaskeleen tukemat ja vaatimat ominaisuudet*

Ominaisuus	Kuvaus	Vaadittu?
name	Askeleen nimi, voidaan esittää käyttöliittymässä.	Kyllä
testFunction	Askeleessa kutsuttava funktio. Funktion tulee löytyä joko sisäänrakennetusta testimoduulista tai testitapauksen määrittämästä testimoduulista.	Ei
parameters	Testifunktiolle annettavat parametrit. Lista parametrien yksilöiviä tunnisteita.	Ei
waitForEvent	Tapahtuman nimi, jonka tapahtumista odotetaan ennen askeleen suorittamista.	Ei
resultAction	Askeleen lopuksi lähetettävän Redux-arkkitehtuurin mukaisen toimenpiteen tyyppi.	Ei
asyncDispatch	Bool-tyyppinen arvo. Määrittää lähetetäänkö resultAction asynkronisesti vai synkronisesti.	Ei, oletusarvo on false

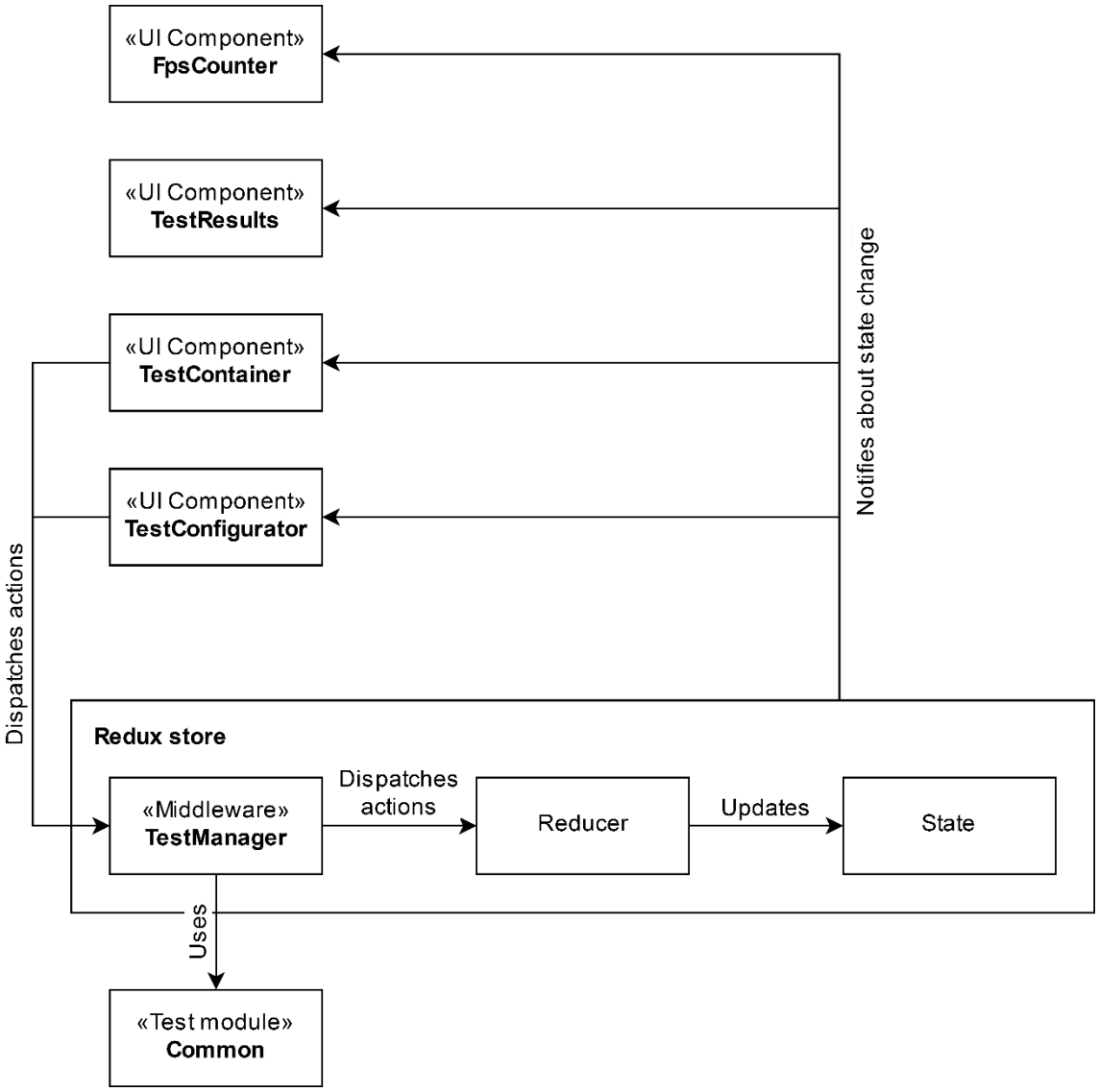
Taulukko 4 *Testiaskelryhmän tukemat ja vaatimat ominaisuudet*

Ominaisuus	Kuvaus	Vaadittu?
name	Askelryhmän nimi, voidaan esittää käyttöliittymässä.	Kyllä
type	Kertoo objektin tyypin.	Kyllä, askelryhmälle arvon tulee olla ”stepGroup”.
initialDelay	Askelryhmän ensimmäistä suorituskertaa edeltävä viive.	Kyllä
intervalParameter	Askelryhmän suoritusintervallin millisekunteina kertovan parametrin nimi.	Kyllä
durationParameter	Askelryhmän suoritusajan millisekunteina kertovan parametrin nimi.	Kyllä
steps	Lista askelryhmässä suoritettavia testiaskelobjekteja.	Kyllä

Diplomityössä käytettävät testitapaukset esitellään tarkemmin aliluvuissa 4.3-4.9.

4.1.4 Testipenkin arkkitehtuuri

Testipenkki toteutettiin Redux-arkkitehtuurin mukaisesti ja React-käyttöliittymäkirjastoa käyttäen. Redux-arkkitehtuuria ei esitellä tässä diplomityössä tarkemmin, vaan lukija voi halutessaan selvittää sen omatoimisesti esimerkiksi Reduxin verkkosivuilta [29]. Kuvassa 8 on esitelty testipenkin keskeisimmät moduulit. Kuvassa nuolet kertovat datan kulkusuunnan, ellei toisin ole mainittu.



Kuva 8. *Testipenkin keskeisimmät moduulit ja datan kulku*

TestConfigurator on React-komponentti, joka tarjoaa kontrollit testattavan komponentin vaihtamiseen, testitapauksen vaihtamiseen, testitapauksen parametrien säätämiseen ja testin käynnistämiseen. Kontrollit näytetään testipenkkiin konfiguroitujen komponenttien ja testitapausten mukaisesti.

TestContainer on React-komponentti, jonka alikomponenttina testattava komponentti on. TestContainer huolehtii testattavan komponentin ja testipenkin välisestä testidatan ja tapahtumien välittämisestä.

TestManager on Redux-middleware, jonka tehtävänä on testitapausten suorittaminen.

Common on testipenkin mukana toimitettava testimoduuli, joka tarjoaa tarvittavat testifunktiot testitapauksia varten.

TestResults ja FpsCounter ovat React-komponentteja, jotka näyttävät nimiensä mukaisesti testitulokset ja visualisoinnin ruudunpäivitysnopeuden kehyksinä sekunnissa.

Reducer ja State ovat Redux-arkkitehtuurin peruspalikoita, jotka huolehtivat sovelluksen tilan muutoksista ja tilasta. Niiden syvällinen ymmärtäminen ei ole oleellista testipenkin ymmärtämisen kannalta.

Kuvassa esiteltyjen moduulien lisäksi testipenkissä on useita muita käyttöliittymäkomponentteja ja vakiomuuttujia sisältäviä moduuleita, joita ei kuvassa esitellä.

4.2 Eri teknologioilla toteutettujen visualisointien mittaustavat

Tässä diplomityössä ei testata muilla teknologioilla tehtyjä visualisointeja. Siitä huolimatta testipenkin suunnittelussa ja testitapausten laatimisessa on pyritty siihen, että testit ovat käyttökelpoisia myös muilla teknologioilla toteutetuille visualisointikomponenteille mahdollisimman vähäisin muutoksin. Toteutettujen testitapausten käyttäminen edellyttää, että visualisointikomponentit toteuttavat luvussa 5 esitetyt vaatimukset visualisointikomponentille.

4.3 Testitapaukset

Seuraavissa aliluvuissa esitellään visualisoinnille suoritettavat testitapaukset. Testitapausten valinnassa on pyritty saavuttamaan joukko, jolla pystytään kattamaan yleisimmät visualisointien käyttötapaukset. Valinnassa on myös pyritty kiinnittämään huomiota testattavuuteen; ei ole mielekästä valita sellaisia testitapauksia, joiden testaaminen on erityisen haastavaa tai mahdotonta.

Jokaiselle testitapaukselle esitellään lyhyet perustelut sen valintaan osaksi suorituskykymittauksia. Lisäksi käydään lävitse testitapauksiin liittyvät testiaskeleet ja testissä mitattavat suureet. Testitapaukset toteutetaan testipenkin avulla, jotta testit ovat helposti toistettavissa useilla laitteilla. Poikkeuksen tähän muodostavat muistin käyttöä ja muistivuotojen havaitsemista koskevat testitapaukset, jotka edellyttävät selaimen kehittäjätyökalujen käyttöä testipenkin lisäksi.

4.4 Visualisoinnin piirtoaika

Oleellinen osa visualisoinnin suorituskykyä on sen piirtämiseen ja lataamiseen kuluva aika. Käytetty aika vaikuttaa suoraan visualisoinnin käyttökokemukseen. Tässä testitapauksessa huomioidaan aika hetkestä, kun visualisointikomponentin piirto

aloitetaan hetkeen, kun visualisointi on käyttökelpoinen. Testi ei huomioi muun käyttöliittymän käytettävyyttä piirtoaikana. JavaScriptin yksisäikeisyyden takia muu sivu saattaa jumiutua raskaan operaation aikana.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 5.

Taulukko 5 *Visualisoinnin piirtoaika -testitapauksen mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Visualisointikomponentin piirtoon kuluva aika	millisekunti	Aikaan sisältyy koko visualisointikomponentin piirto käyttövalmiuteen saakka.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aloita visualisointikomponentin piirtoon kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Piirrä visualisointikomponentti.
- Odota visualisointikomponentilta signaalia piirron valmistumisesta.
- Lopeta visualisointikomponentin piirtoon kuluvan ajan mittaus.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 6.

Taulukko 6 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 1000 datapisteellä
X-akselin suuntainen hajonta	100000
Y-akselin suuntainen hajonta	100000

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.5 Interaktion vasteaika

Interaktiivisen visualisoinnin käyttö edellyttää, että interaktioista seuraava palaute annetaan käyttäjälle mahdollisimman pian käyttäjän suorittaman toimenpiteen jälkeen. Nopea palaute on edellytys hyvälle käyttökokemukselle.

Diplomityössä kehitettävä visualisointikomponentti tukee neljää interaktiota: datapisteen korostaminen hiiren tullessa lähelle tai päälle, datapisteen valitseminen klikkaamalla, visualisoinnin panorointi raahaamalla visualisointia hiirellä ja visualisoinnin zoomausta hiiren rullaa käyttäen. Näiden interaktioiden vasteaikojen mittaaminen antaa käsityksen visualisoinnin interaktiivisuuden käyttökelpoisuudesta.

Interaktioiden testaamista varten testipenkki tarjoaa rajapinnan, jonka kautta hiiren liikkeistä aiheutuvia tapahtumia voidaan luoda ohjelmallisesti ilman hiiren varsinaista liikettä. Nähdään, että tämän lähestymistavan tuoma toistettavuus lisää testitulosten luotettavuutta. Toinen vaihtoehto on liikuttaa hiirtä fyysisesti, mutta tällöin liikkeestä generoituvien tapahtumien määrä saattaa vaihdella ja testin luotettavuus heikkenee. Lisäksi ihmisen tarkkuus on aina konetta heikompi, joten myös inhimillisestä epätarkkuudesta syntyvä virhe vääristäisi testituloksia.

Selkeyden vuoksi kunkin interaktion testaamista varten luodaan oma testitapaus. Näitä testitapauksia käsitellään tarkemmin yksitellen aliluvuissa 4.5.1-4.5.4.

4.5.1 Datapisteen korostaminen hiiren läheisyydestä

Pistekaaviolle on tyypillistä, että yksittäisiä datapisteitä on suuri määrä. Tämän takia käyttäjän voi olla vaikeaa hahmottaa yksittäinen piste muiden joukosta. Yksi keino pisteen havaitsemisen helpottamiseksi on korostaa piste, jonka lähellä tai päällä hiiri on. Jotta tämä apukeino toimii käyttäjäkokemuksen parantamiseksi eikä päinvastoin, tulee korostuksen toimia lähes reaaliaikaisesti hiiren liikkeen mukaisesti.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 7.

Taulukko 7 *Datapisteen korostaminen hiiren läheisyydestä -testitapauksessa mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Datapisteen korostamiseen kuluva aika	millisekunti	Hiiren liikkeen ja datapisteen korostuksen välillä kulunut aika.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.

    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.

- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Aloita datapisteen korostamiseen kuluvan ajan mittaus.
- Liikuta hiiri datapisteen lähelle/päälle.
- Odota visualisointikomponentilta signaalia datapisteen korostamisesta.
- Lopeta datapisteen korostamiseen kuluvan ajan mittaus.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 8.

Taulukko 8 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
X-akselin suuntainen hajonta	50000
Y-akselin suuntainen hajonta	50000
Hiiren sijainti X-akselilla	50 % visualisoinnin leveydestä
Hiiren sijainti Y-akselilla	50 % visualisoinnin korkeudesta

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.5.2 Datapisteen valitseminen klikkaamalla

Datapisteen valitseminen on tyypillinen esimerkki J.S. YI et al. [7] määrittelemien interaktioluokkien ”Valitse” (engl. Select) -luokkaan kuuluvasta interaktiosta. Kuten aiemmin, oleellinen osa interaktion käytettävyyttä on sen vasteaika.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 9.

Taulukko 9 *Datapisteen valitseminen klikkaamalla -testitapauksessa mitattavat suureet*

Suure	**Yksikkö**	**Lyhyt kuvaus**
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Datapisteen valintaan kuluva aika	millisekunti	Hiiren liikkeen ja datapisteen valinnasta kertovan tapahtuman välillä kulunut aika.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Aloita datapisteen valintaan kuluvan ajan mittaus.
- Klikkaa hiirellä annettuun kohtaan.
- Odota visualisointikomponentilta signaalia datapisteen valinnasta.
- Lopeta datapisteen valintaan kuluvan ajan mittaus.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 10.

Taulukko 10 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
X-akselin suuntainen hajonta	50000
Y-akselin suuntainen hajonta	50000
Hiiren sijainti X-akselilla	50 % visualisoinnin leveydestä
Hiiren sijainti Y-akselilla	50 % visualisoinnin korkeudesta

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.5.3 Visualisoinnin panorointi raahaamalla

Visualisoinnin panorointi on tyypillinen esimerkki J.S. Yi et al. [7] määrittelemien interaktioluokkien ”Tutki” (engl. Explore) -luokkaan kuuluvasta interaktiosta. Panoroinnin jatkuvan luonteen takia yksinkertaisen vasteajan sijaan käytettävyydelle oleellista on visualisoinnin ruudunpäivitysnopeus interaktion aikana. Tämän mittaamiseksi testitapauksessa mitataan sekä interaktioon kulunut aika että interaktion aikana piirrettyjen kehysten määrä.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 11.

Taulukko 11 *Visualisoinnin panorointi raahaamalla -testitapauksessa mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Interaktion kesto	sekunti	Panorointiin käytetty aika sekunteina.
Interaktion kehysmäärä	lukumäärä	Panoroinnin aikana piirrettyjen kehysten lukumäärä.
Visualisoinnin keskimääräinen ruudunpäivitystiheys panoroinnin aikana	kehystä sekunnissa, keskiarvo	Interaktion kehysmäärä jaettuna interaktion kestolla.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Aloita interaktioon kuluvan ajan mittaus.
- Aloita interaktion kehysmäärän mittaus.
- Raahaa visualisointia hiirellä.
- Odota onDragEnd-tapahtumaa.
- Lopeta interaktioon kuluvan ajan mittaus.
- Lopeta interaktion kehysmäärän mittaus.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 12.

Taulukko 12 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
X-akselin suuntainen hajonta	100000
Y-akselin suuntainen hajonta	100000
Hiiren alkusijainti X-akselilla	0 % visualisoinnin leveydestä
Hiiren alkusijainti Y-akselilla	0 % visualisoinnin korkeudesta
Hiiren loppusijainti X-akselilla	100 % visualisoinnin leveydestä
Hiiren loppusijainti Y-akselilla	100 % visualisoinnin korkeudesta
Raahausaskeleiden lukumäärä	240

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.5.4 Visualisoinnin zoomaus hiiren rullalla

Visualisoinnin zoomaus kuuluu J.S. Yi et al. [7] mukaan ”Abstrahoi/Selitä” (engl. abstract/elaborate) -interaktioluokkaan. Zoomaus voi olla animoitu, ja täten luonteeltaan jatkuva kuten panorointi, tai tapahtua pykälittäin. Testitapauksen määrittelyssä ei oteta tähän erikseen kantaa. Zoomaukselle mitataan sekä interaktioon kulunut aika että interaktion aikana piirrettyjen kehysten lukumäärä molempien tapausten huomioimiseksi. Tässä testitapauksessa zoomaaminen tapahtuu hiiren rullan käyttöä simuloiden.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 13.

Taulukko 13 *Visualisoinnin zoomaus hiiren rullalla -testitapauksessa mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Interaktion kesto	sekunti	Panorointiin käytetty aika sekunteina.
Interaktion kehysmäärä	lukumäärä	Panoroinnin aikana piirrettyjen kehysten lukumäärä.
Visualisoinnin keskimääräinen ruudunpäivitystiheys panoroinnin aikana	kehystä sekunnissa, keskiarvo	Interaktion kehysmäärä jaettuna interaktion kestolla.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Aloita interaktioon kuluvan ajan mittaus.
- Aloita interaktion kehysmäärän mittaus.
- Pyöritä hiiren rullaa.
- Lopeta interaktioon kuluvan ajan mittaus.

- Lopeta interaktion kehysmäärän mittaus.

Hiiren rullan pyörittäminen tapahtuu askel kerrallansa siten, että kunkin askeleen jälkeen odotetaan visualisoinnilta tietoa zoom-tason muutoksesta ennen seuraavaa rullan askelta. Tarkoituksena on välttää välitön muutos ja täten paremmin simuloida käyttäjän toimintaa.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 14.

Taulukko 14 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
X-akselin suuntainen hajonta	50000
Y-akselin suuntainen hajonta	50000
Zoom-askeleiden lukumäärä	20

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.6 Datan päivittäminen

Visualisoitava data on harvoin staattista. Käyttötapauksen mukaan datan päivitystiheys ja päivitysten luonne voivat vaihdella tunneista millisekunteihin ja koko datasetin vaihdosta pieniin lisäyksiin tai poistoihin. Tässä testitapauksessa keskitytään koko datasetin vaihtoon. Jatkuvaa datan päivittämistä tutkitaan aliluvussa 4.7.

Oleellista tässä testitapauksessa on se, että testattavaa React-komponenttia ei alusteta ja piirretä uudelleen, vaan vain komponentille tarjottava datajoukko vaihtuu. Visualisoinnin päivityksen käytännön toteutus on komponentista riippuvainen; päivitys voi tapahtua piirtämällä koko komponentti uudelleen tai vain muuttuneet osat. Testitapaus ei ota siihen kantaa.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 15.

Taulukko 15 *Datan päivittäminen -testitapauksessa mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Datan piirtoon kuluva aika	millisekunti	Aika datan syöttämisestä visualisoinnin käyttövalmiuteen.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Aloita testidatan generointiin kuluvan ajan mittaus.
- Generoi testidata.
- Lopeta testidatan generointiin kuluvan ajan mittaus.
- Aloita visualisointikomponentin piirtoon kuluvan ajan mittaus.
- Aseta testidata testipenkin tietorakenteeseen.
- Odota visualisointikomponentilta signaalia päivityksen valmistumisesta.
- Lopeta visualisointikomponentin piirtoon kuluvan ajan mittaus.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 16.

Taulukko 16 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
X-akselin suuntainen hajonta	100000
Y-akselin suuntainen hajonta	100000

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.7 Jatkuva datan syöttäminen

Todellisessa käytössä visualisointi päivittyy ja muuttuu jatkuvasti, joten on oleellista perehtyä myös siihen. Tämän työn puitteissa tutkitaan kahta tapausta:

- Koko datajoukko päivittyy minuutin välein.
- Pieniä päivityksiä dataan tehdään 5 sekunnin välein.

Oleellista on visualisoinnin suorituskyvyn säilyminen pidempienkin testiajojen aikana.

Koko datasetin päivittäminen on melko suoraviivaista toteuttaa käyttäen testipenkkiä. Datan generointi ja syöttäminen ajastetaan tapahtumaan minuutin välein, jolloin visualisointikomponentin näkökulmasta mikään ei muutu aiemmin esiteltyihin testitapauksiin nähden.

Pienet päivitykset toteutetaan testipenkin avulla siten, että generoidaan pieni määrä uutta dataa, joka lisätään visualisoinnille aiemmin syötettyyn dataan.

Testissä mitattavat suureet ja niiden kuvaukset on esitelty taulukossa 17. Mittaukset suoritetaan jokaiselle päivitykselle.

Taulukko 17 *Jatkuva datan syöttäminen -testitapauksessa mitattavat suureet*

Suure	Yksikkö	Lyhyt kuvaus
Testidatan generointiin kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona toistojen välillä.
Datan piirtoon kuluva aika	millisekunti	Aika datan syöttämisestä visualisoinnin käyttövalmiuteen.
Lisädatan generointiin ja liittämiseen kuluva aika	millisekunti	Tämän tulisi pysyä lähes vakiona testin aikana.
Lisädatan piirtoon kuluva aika	millisekunti	Aika datan syöttämisestä visualisoinnin käyttövalmiuteen.

Testitapaus koostuu pääpiirteissään seuraavista testiaskeleista:

- Aseta lähtöarvot.
    a. Testiajojen välillä säädetään generoitavien datapisteiden lukumäärää, muuten käytetään samoja lähtöarvoja.
- Suorita minuutin välein.
    a. Aloita testidatan generointiin kuluvan ajan mittaus.
    b. Generoi testidata.
    c. Lopeta testidatan generointiin kuluvan ajan mittaus.
    d. Aloita visualisointikomponentin piirtoon kuluvan ajan mittaus.
    e. Aseta testidata testipenkin tietorakenteeseen.
    f. Odota visualisointikomponentilta signaalia päivityksen valmistumisesta.
    g. Lopeta visualisointikomponentin piirtoon kuluvan ajan mittaus.
- Suorita viiden sekunnin välein.
    a. Aloita testidatan generointiin kuluvan ajan mittaus.
    b. Generoi testidata ja liitä se aiempaan dataan.
    c. Lopeta testidatan generointiin kuluvan ajan mittaus.
    d. Aloita visualisointikomponentin piirtoon kuluvan ajan mittaus.
    e. Aseta testidata testipenkin tietorakenteeseen.
    f. Odota visualisointikomponentilta signaalia päivityksen valmistumisesta.
    g. Lopeta visualisointikomponentin piirtoon kuluvan ajan mittaus.
- Lopeta testi ajan täyttyessä.

Testitapauksessa käytettävät lähtöarvot on esitelty taulukossa 18.

Taulukko 18 *Testitapauksessa käytettävät lähtöarvot*

Parametri	Lähtöarvo
Datapisteiden lukumäärä täydessä päivityksessä	Kasvatetaan testiajojen välillä, aloitetaan 10000 datapisteellä
Uusien datapisteiden lukumäärä pienessä päivityksessä	Kasvatetaan testiajojen välillä, aloitetaan 100 datapisteellä
X-akselin suuntainen hajonta	100000
Y-akselin suuntainen hajonta	100000
Täyden päivityksen intervalli	60000 ms
Pienen päivityksen intervalli	5000 ms
Testin kesto	315000 ms

Taulukosta on jätetty pois testitulosten kannalta epäoleelliset parametrit. Näitä ovat esimerkiksi mittauspisteiden tunnisteet.

Testitapauksen tarkka määrittely (käytetty JSON-tiedosto) on liitteenä.

4.8 Muistin kulutus

Muistin käyttöä ei ole mahdollista seurata testipenkin kautta, sillä selaimet eivät tarjoa tähän soveltuvaa rajapintaa sovellusten käyttöön. Visualisoinnin käyttämän muistin määrä on kuitenkin erittäin merkittävässä asemassa, sillä se vaikuttaa visualisoinnin käyttökelpoisuuteen kevyemmillä laitteilla sekä visualisoinnin jatkokehitysmahdollisuuksiin.

Rajapintojen rajoitteitten takia muistin käyttöä seurataan selainten tarjoamilla kehittäjätyökaluilla. Tässä diplomityössä käytetään Google Chrome 75.0.3770.15 64-bit selainta kehitystyöhön ja testien ajamiseen. Kehittäjätyökalut tarjoavat kattavat mahdollisuudet applikaatioiden muistin ja suorituskyvyn monitorointiin ja analysointiin.

Visualisointikomponentin yksinään käyttämää muistin määrää ei pysty kohtuullisella vaivalla erottelemaan koko testipenkin käyttämästä muistista. Testipenkin yksinään käyttämää muistia voidaan arvioida suorittamalla testejä ilman testattavaa komponenttia, mutta siihen ei tämän työn puitteissa perehdytä.

Oleellista tässä testitapauksessa on selvittää käytetyn muistin määrän suuruusluokka eri datapistemäärillä. Myös varatun muistin absoluuttinen määrä on kiinnostava, sillä eritoten kannettavissa laitteissa on rajallinen määrä muistia käytettävissä. Kerätyistä tuloksista voidaan myös päätellä, miten muistin kulutus käyttäytyy suhteessa datapistemäärään.

Tämä testitapaus vaatii aiempia testitapauksia enemmän manuaalista työtä. Alla on kuvattu tämän testitapauksen testiaskeleet.

- Lataa testipenkki uudelleen.
- Valitse ”Datan päivittäminen”-testitapaus.
- Aseta datapisteiden lukumäärä.
- Avaa kehittäjätyökalut.
- Pakota roskienkeruu.
- Kirjaa ylös käytössä olevan muistin määrä.
- Aja testi.
- Kirjaa ylös käytössä olevan muistin määrä.
- Pakota roskienkeruu.
- Kirjaa ylös käytössä olevan muistin määrä.

Testiä toistetaan, kunnes saavutetaan datapistemäärä, jolla testipenkki kaatuu tai jäätyy. Jäätymisellä tarkoitetaan tilannetta, jossa testipenkki ei vastaa käyttäjän komentoihin minuutin aikana.

4.9 Muistivuoto

Muistivuoto voidaan määritellä muistiksi, jota sovellus ei enää tarvitse, mutta jonka se pitää varattuna siitä huolimatta. Muistivuodolla on useita ei-toivottuja seuraamuksia, kuten epämääräinen hidastuminen, ohjelman kaatuminen ja muille ohjelmille muistin loppumisesta aiheutuvat ongelmat. Vaikka JavaScript onkin varustettu automaattisella muistinhallinnalla, voi muistivuotoja ilmetä ohjelmointivirheiden seurauksena. Pääasiallinen syy muistivuotoihin automaattisella roskienkeruulla varustetuissa ohjelmointikielissä onkin ei-toivotut viitteet. [30]

Muistivuotojen havaitsemisessa selaimen kehittäjätyökalut ovat korvaamaton apu. Google Chrome tarjoaa mahdollisuuden seurata useita suorituskykymittareita aikajanalla, mukaan lukien muistinkäyttöä. Aikajanaa hyödyntäen muistin käytön käyttäytymistä on helppo seurata ja kehittäjätyökalujen avulla roskienkeruun voi pakottaa haluamallaan hetkellä.

Myös tämä testitapaus vaatii aiempia testitapauksia enemmän manuaalista työtä. Muistivuotojen havaitsemiseksi noudatetaan seuraavia testiaskelia.

- Lataa testipenkki uudelleen.
- Valitse ”Jatkuva datan syöttäminen”-testitapaus.
- Aseta sopivat lähtöarvot. Ainakin testiajan tulee olla riittävän pitkä.
- Avaa kehittäjätyökalut.

- Pakota roskienkeruu.
- Aja testi.
- Tyhjennä mittaustulokset testipenkistä.
- Pakota roskienkeruu.
- Vertaile muistin käyttöä aikajanalla testin alussa, aikana ja lopuksi.

Mahdollisten muistivuotojen paikantaminen on osa testitapausta.

5. WEBGL VISUALISOINTIKOMPONENTTI THREE.JS-KIRJASTOLLA

Tässä luvussa käydään lävitse visualisointikomponentin vaatimukset aliluvussa 5.1. Näiden vaatimusten pohjalta päädyttiin toteuttamaan komponentti WebGL-ohjelmointirajapintaa hyödyntäen. WebGL:n valitsemisen tarkemmat perustelut on listattu aliluvussa 5.1.5. Visualisointikomponentin toteutus merkittäviltä osilta esitellään aliluvussa 5.2. Samassa kohdassa perustellaan tehtyjä rajauksia ja rajoituksia.

5.1 Komponentin vaatimukset

Visualisointi toteutetaan React-komponenttina työn tilaajan vaatimuksesta. Muita mahdollisia toteutustapoja olisivat esimerkiksi jonkin toisen käyttöliittymäkehyksen (engl. UI Framework) käyttäminen tai toteuttaminen kokonaan ilman korkeamman tason kapselointia. React-komponentin etuja ovat lopputuloksen helppo käyttöönotto ja jatkokehitys osana jo olemassa olevaa tuotetta.

Toteutettavan komponentin tulee toteuttaa vain itse visualisointi. Datan noutaminen, mahdollinen esikäsittely ja muu suoraan visualisointiin liittymätön toiminnallisuus rajataan toteutettavan komponentin ulkopuolelle.

5.1.1 Komponentin parametrit

React-komponentin parametreja kutsutaan propseiksi. Muuttujien lisäksi komponentille voidaan antaa parametreina myös funktioita. Kehitettävän visualisointikomponentin tulee lähettää toiminnastaan tapahtumia parametrien kautta liitetyille tapahtumakäsittelijäfunktioille (engl. event handler functions). Tapahtumat helpottavat visualisoinnin integroimista osaksi laajempia sovelluksia. Taulukossa 19 on esitelty parametrit, joita visualisointikomponentin tulee tukea.

Taulukko 19 *Toteutettavan visualisointikomponentin tukemat parametrit*

Parametrin nimi	Tietotyyppi	Oletusarvo	Kuvaus
data	[{ x: number, y: number, id: string }]	[]	Visualisoitava data. Parametrin päivitys aiheuttaa visualisoinnin uudelleenpiirtämisen. Listassa voi olla useita objekteja. Objektissa voi olla vaadittujen ominaisuuksien lisäksi muita sarjallistuvia (engl. serializable) ominaisuuksia.
width	number	500	Visualisoinnin leveys pikseleinä.
height	number	300	Visualisoinnin korkeus pikseleinä.
autoWidth	bool	true	true: Visualisointi skaalautuu automaattisesti maksimileveyteen. false: Visualisointi käyttää width-parametrin arvoa leveytenään.
autoHeight	bool	true	true: Visualisointi skaalautuu automaattisesti maksimikorkeuteen. false: Visualisointi käyttää width-parametrin arvoa korkeutenaan.
onReady	Funktio Funktio ei saa parametreja.	-	onReady-tapahtuma lähetetään, kun visualisointi on piirretty ja käyttövalmis. Tapahtuma lähetetään vain kerran komponentin elinkaaren aikana.
onDataClick	Funktio Funktio saa parametrinaan datapisteen, jota klikattiin.	-	onDataClick-tapahtuma lähetetään, kun käyttäjä klikkaa hiirellä visualisoinnissa esitettyä datapistettä.
onDataHover	Funktio Funktio saa parametrinaan datapisteen, jonka päälle hiiri siirtyi.	-	onDataHover-tapahtuma lähetetään, kun käyttäjä vie hiiren visualisoinnissa esitetyn datapisteen päälle.
onDataHoverOut	Funktio Funktio ei saa parametreja	-	onDataHoverOut-tapahtuma lähetetään, kun käyttäjän hiiri poistuu visualisoinnissa esitetyn datapisteen päältä.

onDrag	Funktio Funktio saa parametrinaan raahattua siirtymää kuvaavan objektin.	-	onDrag-tapahtuma lähetetään, kun käyttäjä raahaa visualisointia hiirellään.
onDragStart	Funktio Funktio ei saa parametreja	-	onDragStart-tapahtuma lähetetään, kun käyttäjä aloittaa visualisoinnin panoroinnin hiirellä raahamalla.
onDragEnd	Funktio Funktio ei saa parametreja	-	onDragEnd-tapahtuma lähetetään, kun käyttäjä lopettaa visualisoinnin panoroinnin hiirellä raahamalla.
onFrame	Funktio Funktio ei saa parametreja	-	onFrame-tapahtuma lähetetään aina, kun visualisointikomponentin kehys piirretään.
onMouseMove	Funktio Funktio saa parametrinaan Reactin tuottaman move-tapahtuman	-	onMouseMove-tapahtuma lähetetään, kun käyttäjä liikuttaa hiirtä visualisoinnin päällä.

onZoomLevelChange	Funktio
	Funktio saa parametrinaan uuden zoom-tason kertovan objektin. Esimerkki:
	{
	value: number
	}
		-	onZoomLevelChange-tapahtuma lähetetään, kun visualisoinnin zoom-taso muuttuu.

5.1.2 Komponentin ohjaaminen

Visualisointikomponenttia tulee voida ohjata muusta sovelluksesta. Esimerkiksi visualisointi voidaan haluta keskittää toisessa komponentissa valittuun datapisteeseen. Yksi vaihtoehto olisi toteuttaa toiminnallisuus komponentin parametrien kautta, mutta tällöin komponentista tulisi ohjattu. Ohjatut React-komponentit käyttävät parametreja tilanaan, eikä niitä täten ole mahdollista käyttää ilman ulkopuolista logiikkaa. Toinen vaihtoehto on kutsua komponentin jäsenfunktioita imperatiivisesti käyttäen Reactin viitteitä (engl. refs). Yleisesti ottaen normaali tapa on käyttää parametreja, mutta myös viitteiden käyttö on tuettu [31]. Kehitettävän visualisointikomponentin tapauksessa päädyttiin jäsenfunktioiden käyttämiseen komponentin ohjaamiseen.

Taulukossa 20 on esitetty jäsenfunktiot, jotka visualisointikomponentin tulee toteuttaa.

Taulukko 20 *Visualisointikomponentin tarjoamat jäsenfunktiot*

Funktion nimi	Parametrit	Kuvaus
setZoomLevel(level)	level: number	Asettaa visualisoinnin zoom-tason.
centerVisualizationTo(x, y)	x: number y: number	Keskittää visualisoinnin annettuihin koordinaatteihin.
selectDataPoint(id)	id: string // Datapisteen id	Valitsee datapisteen ja keskittää visualisoinnin valittuun datapisteeseen.

5.1.3 Visualisointityyppi ja tuetut interaktiot

Toteutettavan visualisoinnin tulee olla tyypiltään pistekaavio (engl. scatter plot). Visualisoinnin taustan tulee olla läpinäkyvä, jolloin visualisointia on mahdollista käyttää yhtenä kerroksena osana isompaa visualisointia.

Visualisoinnin tulee tukea taulukossa 21 esiteltyjä interaktioita.

Taulukko 21 *Visualisointikomponentin tukemat interaktiot ja niiden luokat*

Interaktio	Luokka
Datapisteen valinta	Valitse (Select)
Visualisoinnin raahaus	Tutki (Explore)
Visualisoinnin skaalaus (zoom)	Abstrahoi/Selitä (Abstract/Elaborate)

Näiden interaktioiden lisäksi komponentin tulee tukea aliluvussa 5.1.1 esiteltyjä tapahtumia, joiden avulla interaktiota voidaan hyödyntää visualisointikomponentin ulkopuolella.

Työn tilaaja ei asettanut vaatimuksia visualisoinnin ulkoasulle. Pyrkimys on ennen kaikkea selvittää teknisen ratkaisun toimivuutta ja toteuttamiskelpoisuutta. Tästä huolimatta visualisointikomponenttia toteuttaessa tulee pyrkiä huomioimaan ulkoasun jatkokehitysmahdollisuudet.

5.1.4 Laitteisto

Visualisointikomponentin tulee toimia nykyaikaisilla tietokoneilla käyttäen Google Chrome-, Mozilla Firefox- tai Safari-selainta. Tietokoneiden lisäksi visualisoinnin tulee toimia Applen iPad-taulutietokoneella. Aliluvussa 5.1.3 mainittujen interaktioiden tulee toimia sekä hiirellä että kosketuseleillä, joskin suorituskykymittaukset suoritetaan vain hiirtä simuloiden.

Käytettävä laite saa vaikuttaa visualisoinnin sulavuuteen. Vaikutusta pyritään mittamaan osana suorituskykytestejä.

5.1.5 Suorituskyky

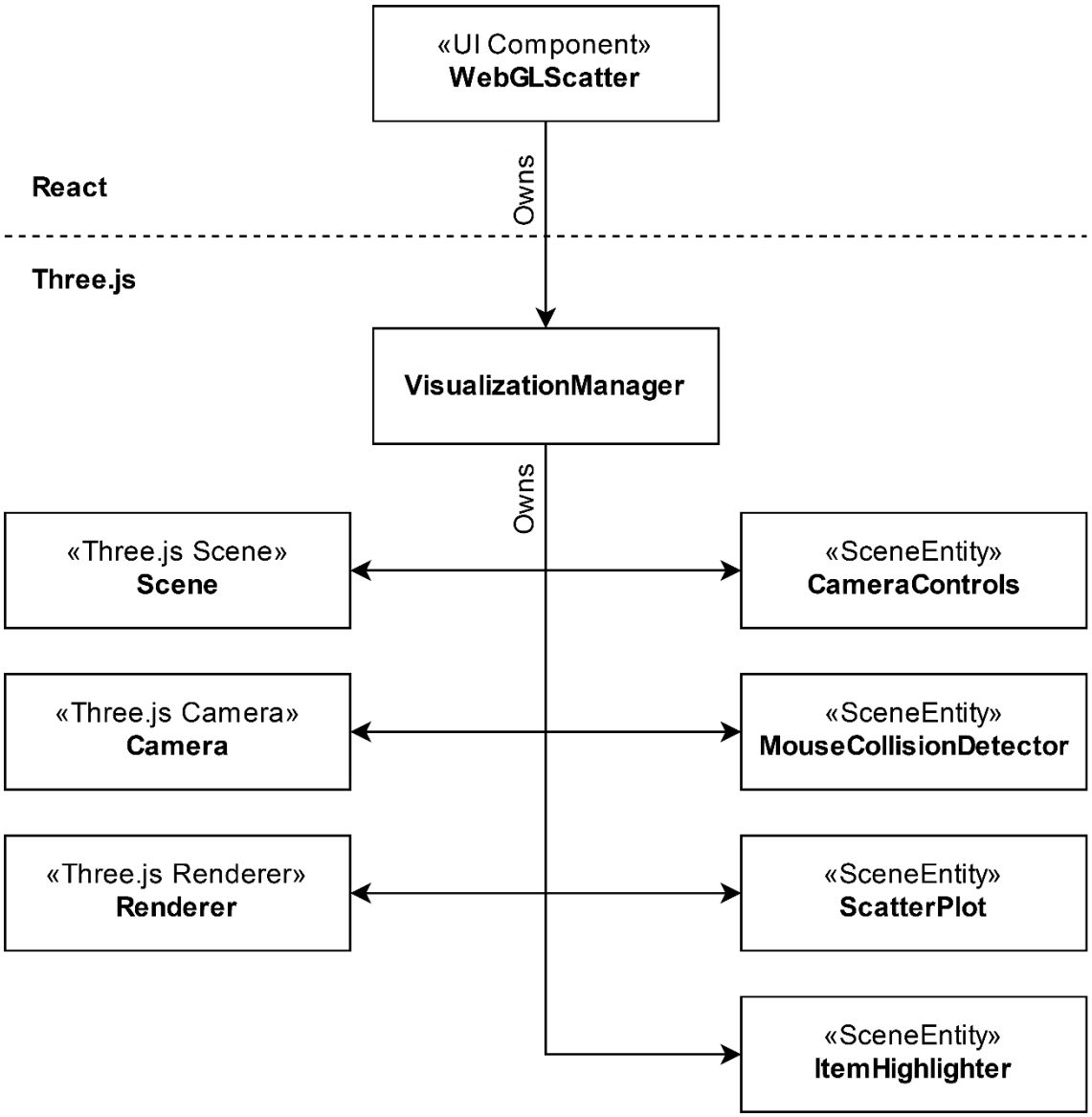
Suoria suorituskykyvaatimuksia ei ole, vaan työssä on tarkoitus selvittää WebGL-pohjaisen ratkaisun suorituskyky. Suorituskyvyn mittaus tulee suorittaa helposti toistettavissa olevalla tavalla, joka on pienellä vaivalla mahdollista suorittaa myös muille visualisoinneille. Visualisointikomponenttia voidaan pitää onnistuneena, mikäli se kykenee esittämään huomattavasti aiempia ratkaisuja suuremman määrän datapisteitä niin, että interaktiot ovat yhä sulavia. Suuntaa-antavana datapistemääränä voidaan pitää 100 000 datapistettä.

WebGL-pohjaisen toteutuksen suorituskyvyn tarkkaan selvittämiseen päädyttiin ennakkotietojen perusteella tehdyn arvion pohjalta. WebGL-rajapinnan suoran käyttämisen sijaan tässä työssä käytetään Three.js-kirjastoa. Näin ollen ohjelmistokehittäjän ei tarvitse itse kirjoittaa käytettäviä varjostinohjelmia, vaan ohjelmointi tapahtuu Three.js:n tarjoaman korkeamman tason rajapintaa hyödyntäen.

Visualisointikomponentin tulee toimia pitkiäkin aikoja lataamatta sivua tai visualisointikomponenttia uudelleen. Minimiedellytys on, että visualisointikomponentti on käyttökelpoinen kahdeksan tunnin ajan, kun visualisoitava data päivittyy viiden minuutin välein.

5.2 Komponentin toteutus

Komponentin toteutus on suunniteltu mahdollisimman modulaariseksi ottaen vaikutteita useista artikkeleista ja foorumiviesteistä. Kuvassa 9 on esitetty diplomityössä toteutetun komponentin rakenne.



Kuva 9. *Visualisointikomponentin moduulit ja korkean tason rakenne*

WebGLScatter on React-komponentti, jonka tehtävänä on tarjota vaadittu rajapinta, luoda canvas DOM-elementti, liittää siihen tarvittavat tapahtumakuuntelijat ja suorittaa visualisoinnin piirtosilmukkaa (engl. render loop). Silmukan jokaisella kierroksella WebGLScatter kutsuu VisualizationManagerin update-metodia ja antaa tälle tarvittavat ohjausarvot, kuten visualisoitavan datan, hiiren liikkeet ja muut tarvittavat tilatiedot.

VisualizationManager huolehtii Three.js-visualisoinnista korkealla tasolla. Tehtäviin kuuluu tarvittavien Three.js-olioiden alustaminen ja näiden hallinta. Näitä olioita ovat Scene-, Camera-, Renderer- ja SceneEntity-oliot. Näistä kolme ensimmäistä alustetaan Three.js:n mukana toimitettavista luokista ja ne tarvitaan, jotta ruudulle pystytään piirtämään mitään. SceneEntity-oliot ovat itse toteutettavia visualisoinnin rakennuspalikoita, eivätkä osa Three.js-kirjastoa.

Scene-olio hallitsee, mitä ja missä visualisoinnissa piirretään. Kaikki visualisoinnissa piirrettävät objektit lisätään Scene-olion hallittavaksi. [32]

Camera-olio nimensä mukaisesti toimii näkymänä visualisointiin. Kameraa liikuttamalla toteutetaan muun muassa zoom- ja panorointi-interaktiot. Three.js tarjoaa useita erilaisia kameroita, kuten esimerkiksi perspektiivi-, ortogonaali- ja stereokameran. Tässä visualisointikomponentissa käytetään perspektiivikameraa, jonka Z-akselin suuntaa ei muuteta missään kohtaa.

Renderer-olio toimii piirtomoottorina. Rendererin tehtävä on piirtää varsinainen kuva annettuun canvas DOM-elementtiin. Three.js tarjoaa WebGLRenderin lisäksi SVG-, CSS2D- ja CSS3DRendererit, joskin vain WebGLRenderer tukee kaikkia Three.js:n ominaisuuksia. Diplomityössä kehitettävässä visualisointikomponentissa käytetään WebGLRendereriä.

SceneEntityt ovat visualisoinnin toiminnallisuuden rakennuspalikoita. Kunkin SceneEntityn vastuulla on selvästi rajattu toiminnallisuus. Uusia SceneEntityjä luomalla visualisointiin on mahdollista lisätä uusia toimintoja muokkaamatta olemassa olevaa koodia.

SceneEntityn rakentajafunktio saa parametrinaan Scene-olion, visualisoinnin tapahtumakuuntelijafunktiot ja tarvittavat konfiguraatioparametrit. Kunkin SceneEntityn tulee tarjota update- ja dispose-metodit.

Update-metodi saa parametrinaan VisualizationManagerin WebGLScatterilta saamat ohjausarvot ja kontekstiobjektin. Kontekstiobjekti sisältää viitteen kameraan, visualisoinnin canvas DOM-elementtiin, visualisoinnin koon pikseleinä ja piirrettävän kehyksen järjestysnumeron. Update-metodia kutsutaan joka kerta, kun uusi kehys piirretään, eli yleensä 60 kertaa sekunnissa.

Dispose-metodia kutsutaan, kun SceneEntityä ollaan poistamassa. Metodin tehtävä on huolehtia, että kaikki SceneEntityn varaamat resurssit vapautetaan ja Scene-oliosta poistetaan SceneEntityn luomat objektit.

CameraControls on SceneEntity, jonka vastuulla on kameran liikuttamisen hallinta. Update-metodissa tarkistetaan kolme tilannetta, joissa kameraa tulee liikuttaa. Nämä ovat visualisoinnin koon muuttuminen, visualisoinnin panorointi ja visualisoinnin zoomaaminen. CameraControls huolehtii kameran liikuttamisen lisäksi tarvittavien tapahtumakuuntelijoiden kutsumisen. Tapahtumakuuntelijoita kutsutaan synkronisesti, jotta mahdolliset visualisointikomponentin ulkopuoliset interaktiot on mahdollista päivittää yhtäaikaisesti visualisoinnin kanssa.

MouseCollisionDetector on SceneEntity, jonka vastuulla on hiiren ja datapisteiden välisten törmäysten havainnointi hiirtä liikuttaessa ja hiiren painiketta painettaessa. Suorituskyvyn parantamiseksi MouseCollisionDetector tarkastelee hiiren liikkeen törmäykset vain joka toisella piirrettävällä kehyksellä. Hiiren klikkaukset tarkastetaan joka kehyksellä. MouseCollisionDetector kutsuu tarvittavia tapahtumakuuntelijoita synkronisesti törmäysten tapahtuessa.

ItemHighlighter on SceneEntity, jonka tehtävä on korostaa annettu datapiste tai datapisteet. ItemHighlighter ei tarkastele hiiren sijaintia, vaan lukee korostettavan datapisteen tai datapisteet update-metodissa saamistaan ohjausarvoista.

ScatterPlot on SceneEntity, jonka tehtävä on lisätä varsinaiset visualisoitavat pisteet Scene-olioon. Update-metodissa tarkistetaan, onko data muuttunut ja tarvittaessa päivitetään piirretyt datapisteet.

Lisäksi kehitystyön aikana on ollut käytössä CoordinateGrid SceneEntity, jonka tehtävänä oli piirtää koordinaatisto visualisoinnin pohjalle.

Valitun ratkaisun lisäksi pohdittiin React Three Fiber-paketin käyttöä visualisoinnin osien hallintaan. React Three Fiber on React-renderer, jonka avulla React pystyy hallinnoimaan Three.js-sceneä vastaavasti kuin DOMia. Lopullinen syy kallistua valittuun ratkaisuun oli visualisointikomponentin sisäisen React-riippuvuuden välttäminen ja visualisoinnin suorituskykyyn mahdollisesti vaikuttavien muuttujien eliminointi.

5.2.1 Tapahtumien käsittely

Oleellinen osa visualisointikomponentin interaktiivisuutta on tapahtumien mielekäs käsittely. Tapahtumilla tarkoitetaan tässä yhteydessä käyttäjän aiheuttamia tapahtumia, kuten hiiren ja näppäimistön käyttöä. Jotta käyttäjä voi hallita visualisointia hiiren avulla, tulee hiireen liikkeistä syntyvät tapahtumat käsitellä järkevällä ja tehokkaalla tavalla.

Visualisointikomponentin ylimmän tason moduuli, WebGLScatter, liittää tapahtumakuuntelijat DOM-elementtiin seuraaville interaktioille:

- onclick (klikkaus)
- onmousemove (hiiren liike)
- onwheel (hiiren rulla)
- onmousedown (hiiren painikkeen painaminen)
- onmouseup (hiiren painikkeen nostaminen)
- ontouchstart (kosketuksen aloitus)
- ontouchend (kosketuksen lopetus)

- ontouchmove (kosketuksen liike).

Tapahtumien käsittely tapahtuu asynkronisesti, joten tapahtumaa käsitellessä ei ole tietoa piirtosilmukan vaiheesta. On mahdollista, että visualisoinnin kehysten välillä tapahtuu esimerkiksi useita hiiren klikkauksia. Tapahtumakäsittelijöiden tulee huomioida tämä päivittäessään visualisoinnin ohjausarvoja. Piirtosilmukka lukee ohjausarvot komponentin tilasta ja nollaa tilan lukemisen jälkeen.

Kosketustapahtumien käsittelyssä vaatii erityishuomiota se, että siinä missä hiiriä on vain yksi, voi kosketusnäytöllä olla yhtä aikaa useita sormia. Toinen erityispiirre on hiiren rulla, mitä ei sormista löydy, vaan visualisoinnin zoomausta varten tuetaan yleisen konvention mukaisesti kahden tai useamman sormen nipistyselettä.

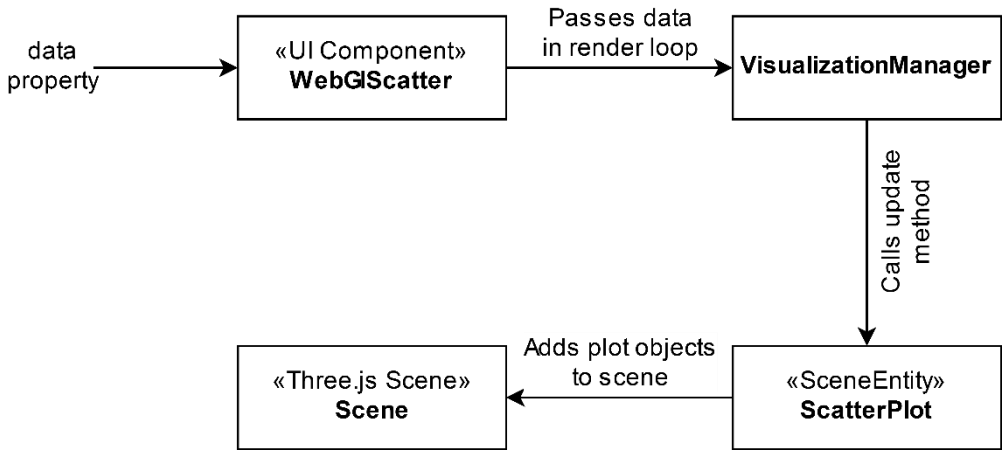
WebGLScatter antaa VisualizationManagerille listan visualisoinnin tapahtumakuuntelijoita, jotka kuuntelevat visualisoinnissa tapahtuvia tapahtumia. Näitä ovat esimerkiksi datapisteiden ja hiirten törmäykset sekä datapisteiden valinta. Visualisoinnin tapahtumakuuntelijat vuorostaan kutsuvat WebGLScatterille parametreina annettuja tapahtumakuuntelijoita.

Visualisointikomponentin integroiminen osaksi laajempaa visualisointia vaatii, että visualisoinnin lävitse on mahdollista klikata. Tämän toteuttaminen vaatii lisälogiikkaa MouseCollisionDetectoriin, sillä WebGLScatterin tasolla ei ole tietoa datapisteiden absoluuttisesta sijainnista näytöllä. Diplomityössä ei toteutettu tukea visualisoinnin lävitse tapahtumien välittämiselle, mutta tuen lisääminen ei vaadi merkittävää lisätyötä.

Visualisointikomponentti kokonaisuudessaan kuuntelee käyttäjän toimista aiheutuvia tapahtumia ja lähettää uusia tapahtumia visualisoinnissa tapahtuvien tapahtumien perusteella.

5.2.2 Datan vastaanotto ja piirto

Visualisoinnin pääasiallinen tehtävä on visualisoida dataa ja täten helpottaa datan ymmärtämistä. Visualisointikomponentille visualisoitava data annetaan data-parametrissa. Kuvassa 10 on esitetty visualisoitavan datan kulkeutuminen visualisointikomponentissa.



Kuva 10. *Visualisoitavan datan reitti visualisointikomponentissa*

Data kulkee muiden ohjausarvojen mukana piirtosilmukassa WebGLScatter React-komponentista VisualizationManagerille, joka puolestaan välittää datan ohjausarvojen mukana SceneEntityille. SceneEntityjen joukossa on ScatterPlot, joka vastaanottaa datan ja luo sen perusteella Three.js-datapisteobjektit. Ennen datapisteobjektien luomista ScatterPlot tarkastaa, onko data muuttunut vertailemalla datamuuttujan viitettä edellisellä päivityksellä saatuun datamuuttujan viitteeseen. Näin pystytään välttämään datan turha uudelleenpiirtäminen.

Mikäli data on muuttunut päivitysten välissä, ScatterPlot siivoaa ensin vanhat datapisteet pois Scene-oliosta ja tämän jälkeen luo uudet datapisteobjektit muuttuneen datan perusteella. Toinen vaihtoehto olisi hyödyntää jo aiemmin luotuja datapisteobjekteja, ja täten välttyä turhalta poistamiselta ja uudelleen luomiselta. Tähän ei kuitenkaan tämän diplomityön puitteissa ryhdytty. Suorituskykytestien perusteella arvioidaan uudelleen, onko tälle tarvetta.

6. SUORITUSKYKYTESTIT

Tässä luvussa esitellään suorituskykytesteihin käytetty laitteisto, käydään lävitse suorituskykytestien tulokset ja käsitellään saatuja tuloksia lyhyesti. Laitteisto esitellään aliluvussa 6.1. Tulosten yhteenvedot on jaettu alilukuihin 6.2-6.7 luvussa 4 esiteltyjen testitapausten mukaisesti. Yksityiskohtaisia mittaustuloksia ei ole diplomityön liitteenä niiden suurehkon määrän vuoksi (yli 3000 riviä), mutta ovat saatavilla pyynnöstä.

Testit on ajettu selain kokoruututilassa ja sivu on päivitetty jokaisen testiajon jälkeen, ellei toisin ole mainittu.

6.1 Laitteisto

Suorituskykytestit suoritettiin tehokkaalla pöytätietokoneella, vanhalla Apple MacBook Prolla ja uudella Apple iPad Prolla. Tarkat tekniset tiedot käytetyille laitteille ovat esitelty taulukoissa 22, 23 ja 24.

Taulukko 22 *Tehokkaan pöytätietokoneen tekniset tiedot*

Komponentti	Arvo
Näytönohjain	MSI GTX1080 8 Gt
Prosessori	Intel i5-4670K, 4-ydintä, kellotaajuus 4,3 GHz
Keskusmuisti	16 Gt DDR3 1600 MHz
Näytön resoluutio	3440 x 1440
Selain	Google Chrome 75.0.3770.15 64-bit

Taulukko 23 *Apple MacBook Pro mid 2009 tekniset tiedot*

Komponentti	Arvo
Näytönohjain	NVIDIA GeForce 9400M 256 Mt
Prosessori	2,26 GHz Intel Core 2 Duo
Keskusmuisti	4 Gt DDR3 1067 MHz
Näytön resoluutio	1280 x 800
Selain	Mozilla Firefox Developer Edition 67.0b15 64-bit

Selaimena MacBook Prolla käytettiin Firefoxia, koska vain se tuki WebGL:ä käytössä olevan näytönohjaimen kanssa.

Taulukko 24 *Apple iPad Pro 12,9” 256 Gt*

Komponentti	Arvo
SoC	Apple A12X Bionic
Keskusmuisti	4 Gt LPDDR4X
Näytön resoluutio	2732 x 2048
Selain	Safari 12 (iOS 12.2)

6.2 Visualisoinnin piirtoaika

Testitapauksessa käytettiin datapistemäärinä arvoja 1 000 – 6 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Taulukossa 25 on esitelty keskimääräinen visualisoinnin piirtoaika jokaiselle testatulle yhdistelmälle.

Taulukko 25 *Keskimääräinen visualisoinnin piirtoaika millisekunteina per datapistemäärä ja laite*

Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
1 000	308	751	222
10 000	358	745	236
100 000	376	862	253
1 000 000	624	1554	502
2 000 000	1450	2093	812
3 000 000	2806	3296	1274
4 000 000	4802	3926	
5 000 000	6130	5303	
6 000 000		20337	

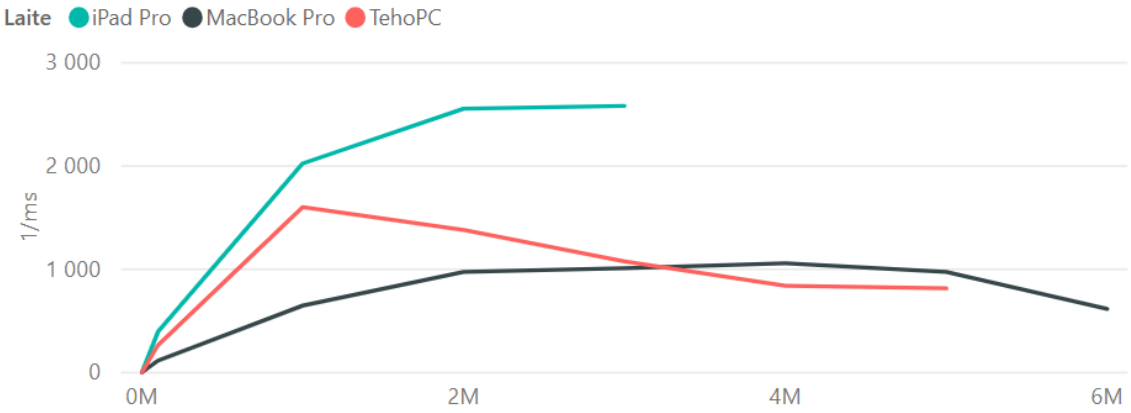
TehoPC:llä 5 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kahdella toistolla viidestä. iPad Pro:lla 3 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 4 000 000 datapisteellä selain ilmoitti virheestä. MacBook Prolla testitapauksen ajo sujui ongelmitta, mutta 6 000 000 datapistettä aiheutti selvän piikin visualisoinnin piirtoaikaan.

Ongelmat johtuvat oletettavasti muistin loppumisesta. TehoPC:llä selaimen muistinkäyttöä tarkasteltaessa havaitaan, että muistinkäyttö on noin 4 Gt kun välilehti kaatuu. Google Chrome käyttää JavaScriptin suorittamiseen V8-moottoria, joka oletusasetuksilla sallii maksimissaan 4 Gt muistinkäytön. Kun tämä havainto yhdistetään tietoon siitä, että iPad Prossa on 4 Gt keskusmuistia yhteensä, voidaan muistin loppumista pitää melko varmana syynä ongelmiin.

MacBook Prolla selaimena on käytössä Mozilla Firefox, joka käyttää JavaScriptin suorittamiseen SpiderMonkey-moottoria. Todennäköisesti se ei aseta vastaavia

rajoituksia muistin käytölle, sillä suuretkin datapistemäärät toimivat TehoPC:llä Mozilla Firefoxia kokeiltaessa. MacBook Pron merkittävä hidastuminen 6 000 000 datapisteellä johtunee keskusmuistin loppuessa käytettävästä sivutustiedostosta.

Kuvassa 11 on esitelty datapistemäärä jaettuna visualisoinnin piirtoajalla.



Kuva 11. *Datapistemäärä jaettuna visualisoinnin piirtoajalla per laite datapistemäärän funktiona*

Kuvasta havaitaan selvästi, että iPad Pron suorituskyky on testatuista laitteista paras. TehoPC:n suorituskyky laskee jo 1 000 000 datapisteen jälkeen painuen MacBook Prota kehnommaksi.

6.3 Datapisteen korostaminen hiiren läheisyydestä

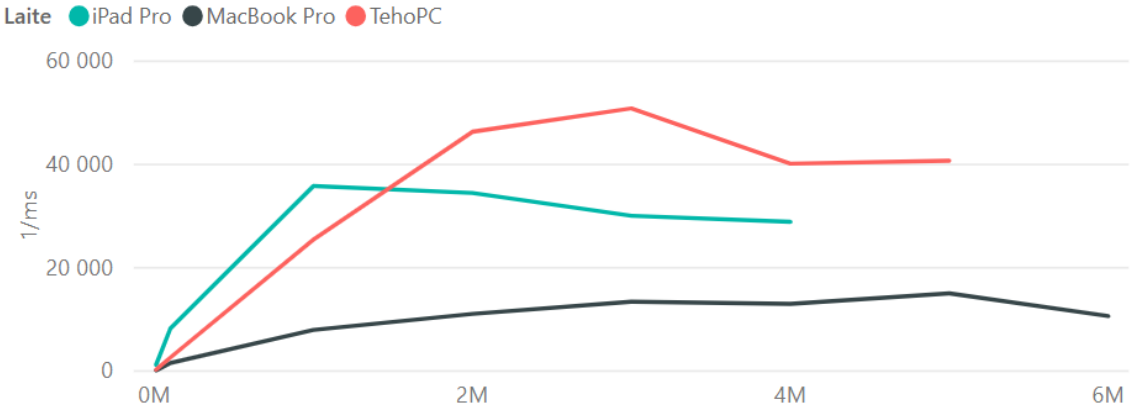
Testitapauksessa käytettiin datapistemäärinä arvoja 10 000 – 6 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Tuloksia arvioitaessa tulee muistaa, että datapisteen korostamisen laskenta suoritetaan vain joka toisella piirrettävällä kehyksellä. Taulukossa 26 on esitelty keskimääräinen datapisteen korostamiseen kuluva aika jokaiselle testatulle yhdistelmälle.

Taulukko 26 *Keskimääräinen datapisteen korostamiseen kuluva aika millisekunteina per datapistemäärä ja laite*

Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
10 000	37	84	9
100 000	42	65	13
1 000 000	43	127	29
2 000 000	45	181	62
3 000 000	59	226	102
4 000 000	103	349	139
5 000 000	124	346	
6 000 000		746	

TehoPC:llä 5 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kahdella toistolla viidestä. iPad Prolla 4 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 5 000 000 datapisteellä selain ilmoitti virheestä. Huomionarvoista on, että aiemmassa testitapauksessa iPad Pro suoriutui vain 3 000 000 datapisteestä. Testitapausten välissä visualisointiin tai testipenkkiin ei ole tehty muutoksia eikä iPad Prota ole päivitetty. MacBook Prolla testitapauksen ajo sujui ongelmitta, joskin 6 000 000 datapisteellä testitapauksen kokonaissuoritusaika kasvoi havaittavasti.

Alla olevassa kuvassa 12 on esitelty datapistemäärä jaettuna datapisteen korostamiseen kuluvalla ajalla.



Kuva 12. *Datapistemäärä jaettuna datapisteen korostamiseen kuluvalla ajalla per laite datapistemäärän funktiona*

Kuvasta havaitaan, että TehoPC:n suorituskyky on testatuista laitteista paras, kun datapistemäärä kasvaa ja MacBook Pron kehnoin. 1 000 000 datapisteeseen asti iPad Pro on hieman TehoPC:tä suorituskykyisempi, joskin taulukon 26 lukuarvoja vertailemalla havaitaan, että erot millisekunteina ovat melko pieniä.

6.4 Datapisteen valitseminen klikkaamalla

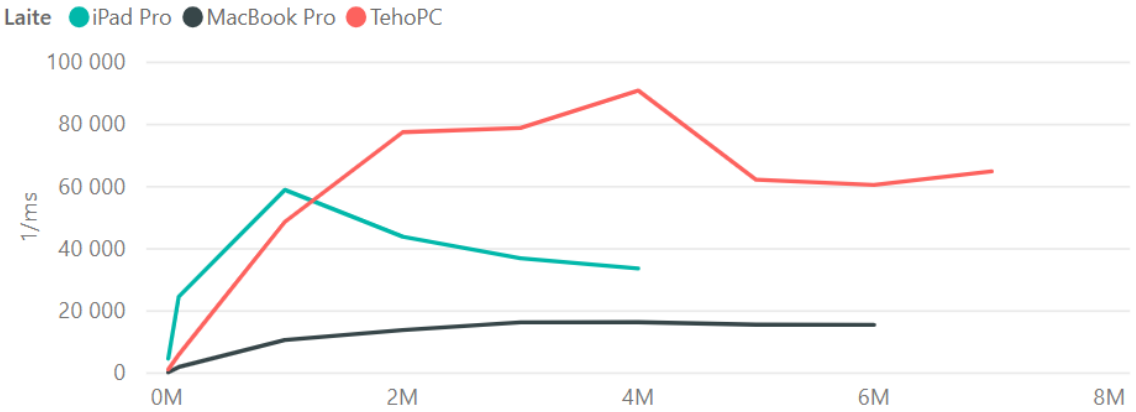
Testitapauksessa käytettiin datapistemäärinä arvoja 10 000 – 7 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Taulukossa 27 on esitelty keskimääräinen datapisteen valintaan kuluva aika jokaiselle testatulle yhdistelmälle.

Taulukko 27 *Keskimääräinen datapisteen valintaan kuluva aika millisekunteina per datapistemäärä ja laite*

Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
10 000	8	41	2
100 000	17	52	6
1 000 000	21	94	18
2 000 000	26	145	50
3 000 000	38	184	84
4 000 000	45	247	119
5 000 000	94	347	
6 000 000	100	400	
7 000 000	108		

TehoPC:llä 7 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kolmella toistolla viidestä. Aikaisemmissa testitapauksissa jo 5 000 000 datapistettä on aiheuttanut ongelmia. Muutokselle ei ole selvää syytä, sillä sekä käytetty laitteisto että ohjelmisto ovat pysyneet muuttumattomina. iPad Prolla 4 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 5 000 000 datapisteellä selain ilmoitti virheestä. MacBook Prolla testitapauksen ajo sujui ongelmitta.

Alla olevassa kuvassa 13 on esitetty datapistemäärä jaettuna datapisteen valintaan kuluvalla ajalla.



Kuva 13. *Datapistemäärä jaettuna datapisteen valintaan kuluvalla ajalla per laite datapistemäärän funktiona*

Kuvasta havaitaan, että laitteiden välinen suorituskyky käyttäytyy kuten edellisen luvun datapisteen korostamista käsittelevässä testitapauksessa. Tulos on odotettu, sillä datapisteen korostaminen ja valinta hiirellä ovat keskenään samankaltaisia operaatioita.

6.5 Visualisoinnin panorointi raahaamalla

Testitapauksessa käytettiin datapistemäärinä arvoja 10 000 – 6 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Taulukossa 28 on esitelty keskimääräinen ruudunpäivitysnopeus jokaiselle testatulle yhdistelmälle. Keskimääräinen panorointiin kulunut aika jokaiselle testatulle yhdistelmälle on esitelty taulukossa 29.

Taulukko 28 *Keskimääräinen ruudunpäivitysnopeus kehyksinä sekunnissa per datapistemäärä ja laite*

Datapistemäärä	TehoPC (fps)	MacBook Pro (fps)	iPad Pro (fps)
10 000	60	57,9	60,2
100 000	59,6	43,1	60,3
1 000 000	59,8	28	60,3
2 000 000	59,9	19,5	59,9
3 000 000	59,5	14,9	60,3
4 000 000	59	12,4	45
5 000 000	58,4	10,7	
6 000 000	57,6	9,5	

Taulukko 29 *Keskimääräinen panorointiin kulunut aika millisekunteina per datapistemäärä ja laite*

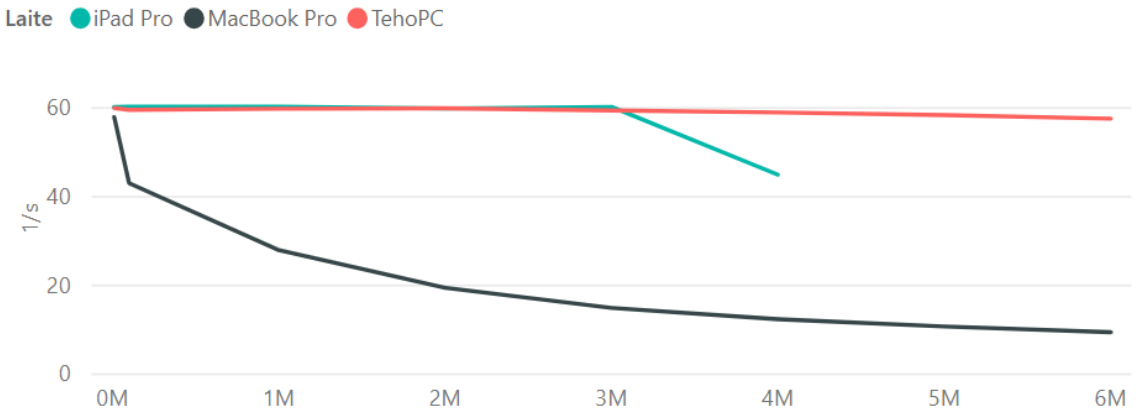
Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
10000	3836	3849	3843
100000	3861	3855	3849
1000000	3847	3863	3842
2000000	3841	3922	3864
3000000	3849	3954	3834
4000000	3883	4003	3849
5000000	3897	4027	
6000000	3897	3970	

TehoPC:llä 6 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kolmella toistolla viidestä. iPad Prolla 4 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 5 000 000 datapisteellä selain ilmoitti virheestä. MacBook Prolla testitapauksen ajo sujui ongelmitta.

Taulukon 28 tuloksista havaitaan, että TehoPC selviytyi suuristakin datamääristä ongelmitta. iPad Pro:n tulos notkahti vain hieman 4 000 000 datapisteellä. MacBook Pron ruudunpäivitysnopeus laski nopeasti datapistemäärän kasvaessa ja jäi täten selvästi muita heikommaksi.

Taulukon 29 tuloksista nähdään, että panorointiin kulunut aika pysyi lähes vakiona, mikä on testitapauksen toteutustavan perusteella olettavissa. Testitapauksessa hiirellä raahaamista simuloivat tapahtumat lähetettiin visualisointikomponentille tasaisin intervallein, jolloin yhden kehyksen aikana saatettiin prosessoida useita raahaustapahtumia. Poikkeaman vakioaikaan tuo MacBook Pro, jolla aika hieman kasvoi suurilla datapistemäärillä. Poikkeaman selittää laitteen heikko suorituskyky panoroinnin aikana, kuten taulukon 26 tuloksista havaitaan. Laitteen ollessa kovassa rasituksessa tapahtumien lähettämisen intervallit saattavat olla odotettua pidempiä.

Alla olevassa kuvassa 14 on esitelty taulukon 28 arvot kuvaajana.



Kuva 14. *Keskimääräinen ruudunpäivitysnopeus kehyksinä sekunnissa per laite datapistemäärän funktiona*

6.6 Visualisoinnin zoomaus hiiren rullalla

Testitapauksessa käytettiin datapistemäärinä arvoja 10 000 – 6 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Taulukossa 30 on esitelty keskimääräinen ruudunpäivitysnopeus jokaiselle testatulle yhdistelmälle. Keskimääräinen zoomaukseen kulunut aika jokaiselle testatulle yhdistelmälle on esitelty taulukossa 31.

Taulukko 30 *Keskimääräinen ruudunpäivitysnopeus kehyksinä sekunnissa per datapistemäärä ja laite*

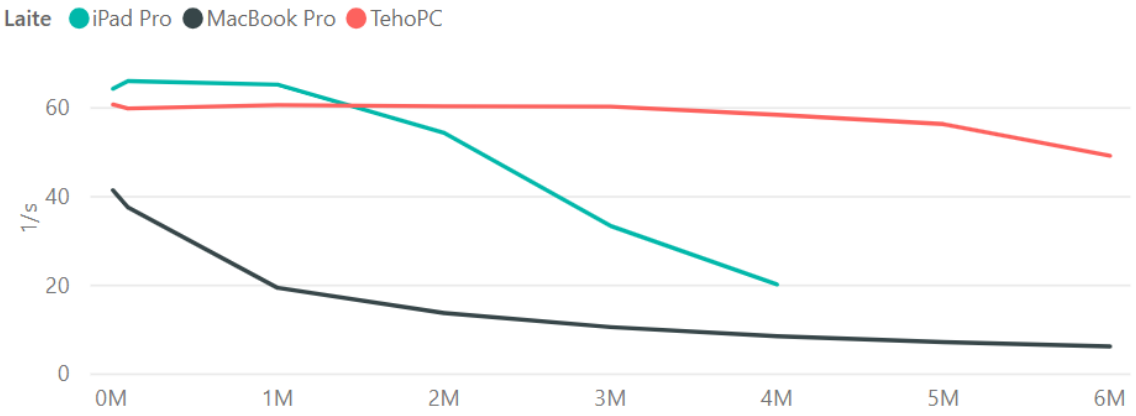
Datapistemäärä	TehoPC (fps)	MacBook Pro (fps)	iPad Pro (fps)
10 000	60,8	41,5	64,4
100 000	59,9	37,6	66,1
1 000 000	60,7	19,5	65,3
2 000 000	60,4	13,8	64,4
3 000 000	60,3	10,6	33,4
4 000 000	58,5	8,6	20,2
5 000 000	56,4	7,2	
6 000 000	49,2	6,3	

Taulukko 31 *Keskimääräinen zoomaukseen kulunut aika millisekunteina per datapistemäärä ja laite*

Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
10 000	329	486	311
100 000	334	533	303
1 000 000	330	1040	306
2 000 000	331	1456	371
3 000 000	332	1884	599
4 000 000	343	2337	990
5 000 000	356	2765	
6 000 000	407	3194	

TehoPC:llä 6 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kolmella toistolla viidestä. iPad Prolla 4 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 5 000 000 datapisteellä selain ilmoitti virheestä. MacBook Prolla testitapauksen ajo sujui ongelmitta.

Alla olevassa kuvassa 15 on esitelty taulukon 30 arvot kuvaajana.



Kuva 15. *Keskimääräinen ruudunpäivitysnopeus kehyksinä sekunnissa per laite datapistemäärän funktiona*

Taulukon 30 ja kuvan 15 perusteella havaitaan, että TehoPC:n suorituskyky oli testatuista laitteista paras ja MacBook Pron huonoin. iPad Pron alun yli 60 kehystä sekunnissa nopeudelle ei ole varmaa syytä, sillä vaikka laitteen näyttöpaneeli tukeekin 120 Hz virkistystaajuutta, nettiselain on rajoitettu 60 kehystä sekunnissa nopeuteen.

Taulukon 31 tuloksista havaitaan, että zoomaukseen kulunut aika ei ole pysynyt vakiona. Tälle selitys on testitapauksen toteutustapa, jossa hiiren rullan liikautusta simuloiva tapahtuma lähetettiin vasta edellisen zoomauksen toteuduttua. Tällä pyrittiin siihen, että visualisointikomponentti joutuu prosessoimaan joka tilanteessa yhtä monta kameran liikuttamista ja täten tuomaan mahdolliset suorituskykyerot selvästi esiin.

6.7 Datan päivittäminen

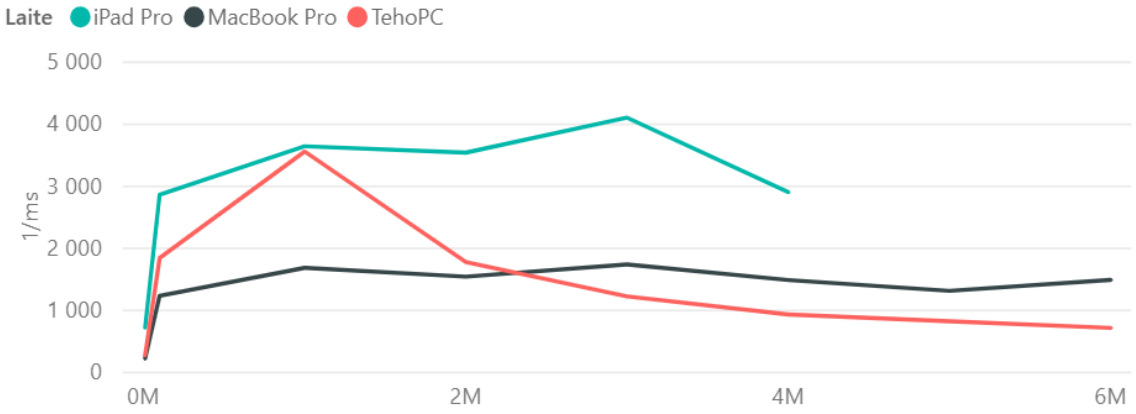
Testitapauksessa käytettiin datapistemäärinä arvoja 10 000 – 6 000 000. Testitapaus toistettiin jokaisella laite- ja datapistemääräyhdistelmällä viisi kertaa. Taulukossa 32 esitelty keskimääräinen datan päivittämiseen kulunut aika jokaiselle testatulle yhdistelmälle.

Taulukko 32 *Keskimääräinen datan päivittämiseen kulunut aika millisekunteina per datapistemäärä ja laite*

Datapistemäärä	TehoPC (ms)	MacBook Pro (ms)	iPad Pro (ms)
10 000	38	46	15
100 000	55	88	35
1 000 000	281	593	277
2 000 000	1140	1323	574
3 000 000	2479	1723	733
4 000 000	4317	2722	1383
5 000 000	6054	3798	
6 000 000	8350	4078	

TehoPC:llä 6 000 000 datapistettä aiheutti ongelmia ja selaimen välilehti kaatui kahdella toistolla viidestä. iPad Prolla 4 000 000 datapisteellä testitapauksen suorittaminen sujui ongelmitta, mutta 5 000 000 datapisteellä selain ilmoitti virheestä. MacBook Prolla testitapauksen ajo sujui ongelmitta.

Alla olevassa kuvassa 16 on esitetty datapistemäärä jaettuna datan päivittämiseen kuluneella ajalla.



Kuva 16. *Datapistemäärä jaettuna datan päivittämiseen kuluneella ajalla datapistemäärän funktiona*

Kuvasta 16 havaitaan, että laitteiden suhteellinen suorituskyky toisiinsa nähden käyttäytyy melko vastaavasti kuin visualisoinnin piirtoaikaa mitattaessa. iPad Pro suoriutuu parhaiten, mutta ei kykene käsittelemään yhtä suuria datapistemääriä kuin TehoPC ja MacBook Pro. TehoPC pärjää alkuun hyvin, mutta suuremmilla datapistemäärillä jää MacBook Pron jalkoihin.

6.8 Jatkuva datan syöttäminen

Testitapauksessa käytetyt datapistemääräkombinaatiot on esitelty taulukossa 33. Jokaisella datapistemäärä kombinaatiolla suoritettiin yksi toisto jokaisella laitteella, poikkeuksena taulukon alimmainen kombinaatio, jota ei pystytty iPad Prolla suorittamaan.

Luettavuuden parantamiseksi käytetään datapistemäärästä täydessä päivityksessä lyhennettä DPTP ja datapistemäärästä pienessä päivityksessä DPIP tässä luvussa esiteltävissä kaavioissa.

Minuutin välein tapahtuvassa täydessä päivityksessä visualisointikomponentin sillä hetkellä esittämä data korvataan DPTP:n määrittämällä määrällä uusia datapisteitä. Viiden sekunnin välein tapahtuvassa pienessä päivityksessä visualisointikomponentin esittämään dataan lisätään DPIP:n määrittämä määrä uusia datapisteitä.

Taulukko 33 *Jatkuva datan syöttäminen -testitapauksessa testatut datapistemääräkombinaatiot*

Datapistemäärä täydessä päivityksessä (DPTP)	Datapistemäärä pienessä päivityksessä (DPIP)
10 000	100
100 000	100
100 000	1 000
100 000	10 000
1 000 000	1 000
1 000 000	10 000
1 000 000	100 000

Taulukossa 34 on esitelty keskimääräinen pieneen päivitykseen kulunut aika (taulukossa ”Lisäys”) millisekunteina ja keskimääräinen täyteen päivitykseen kulunut aika millisekunteina testin aikana iPad Prolla. Taulukossa 35 on esitelty vastaavat arvot MacBook Prolle ja taulukossa 36 TehoPC:lle.

Taulukko 34 *Keskimääräinen pieneen ja täyteen päivitykseen kulunut aika millisekunteina testatuilla datapistemääräkombinaatioilla iPad Prolla*

	Pieni lisäys	**Täysi päivitys**	**Pieni lisäys**	**Täysi päivitys**	**Pieni lisäys**	**Täysi päivitys**
DPTP / DPIP	**10 000**	**10 000**	**100 000**	**100 000**	**1 000 000**	**1 000 000**
100	36	14	49	36		
1 000			33	36	35	223
10 000			40	32	68	150

Taulukko 35 *Keskimääräinen pieneen ja täyteen päivitykseen kulunut aika millisekunteina testatuilla datapistemääräkombinaatioilla MacBook Prolla*

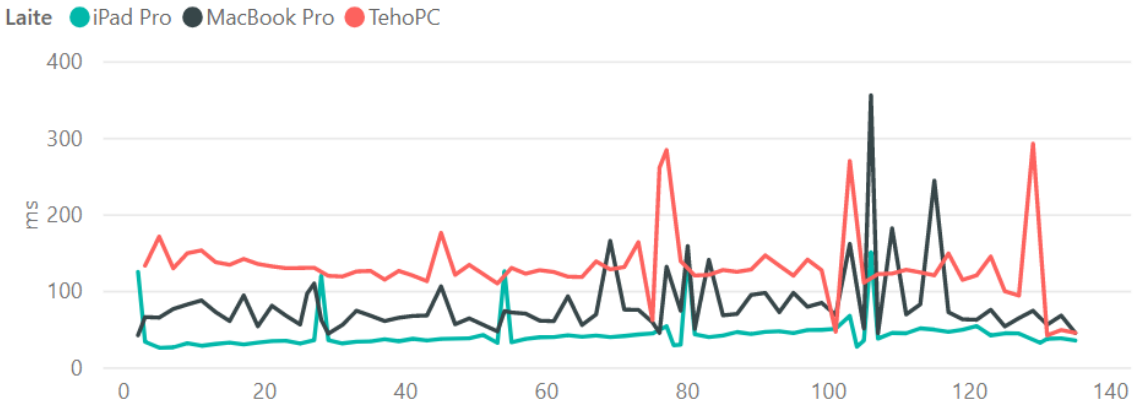
	Pieni lisäys	**Täysi päivitys**	**Pieni lisäys**	**Täysi päivitys**	**Pieni lisäys**	**Täysi päivitys**
DPTP / DPIP	**10 000**	**10 000**	**100 000**	**100 000**	**1 000 000**	**1 000 000**
100	28	35	67	96		
1 000			64	81	38	56
10 000			113	106	194	362
100 000					1722	1479

Taulukko 36 *Keskimääräinen pieneen ja täyteen päivitykseen kulunut aika millisekunteina testatuilla datapistemääräkombinaatioilla TehoPC:llä*

	Pieni lisäys	Täysi päivitys	Pieni lisäys	Täysi päivitys	Pieni lisäys	Täysi päivitys
DPIP \ DPTP	10 000	10 000	100 000	100 000	1 000 000	1 000 000
100	46	28	50	45		
1 000			49	47	287	263
10 000			52	43	302	264
100 000					665	392

Taulukoiden 34-36 tulokset ovat pääpiirteittäin linjassa aiempien testitapauksien kanssa. Eräs mielenkiintoinen poikkeama on MacBook Pron täyden datan päivityksen erityisen hyvä tulos, 56 ms, kun DPTP on 1 000 000 ja DPIP 1000. Datan päivittäminen -testitapauksessa vastaavan tilanteen tulos MacBook Prolla on 593 ms, joka on merkittävästi heikompi (katso taulukko 32). Datan päivittäminen testitapaus vastaa kuormitukseltaan yksittäistä täyttä päivitystä.

Tämän testitapauksen tarkoitus oli selvittää, kuinka visualisointikomponentin suorituskyky säilyy hieman pidemmän käytön aikana. Käyttäytymisen arvioimiseksi tarkastellaan keskiarvojen lisäksi pienen päivityksen kestoa mittauspisteen järjestysluvun funktiona alla olevan kuvan 17 avulla. Kuvassa käytetystä mittausaineistosta on suodatettu pois rivit, joilla DPIP on 100 000, jotta kaikilla laitteilla on suoritettuna vastaavat testit.



Kuva 17. *Keskimääräinen pieneen päivitykseen kulunut aika millisekunteina per laite mittauspisteen järjestysluvun funktiona (DPIP 100 000 on suodatettu pois)*

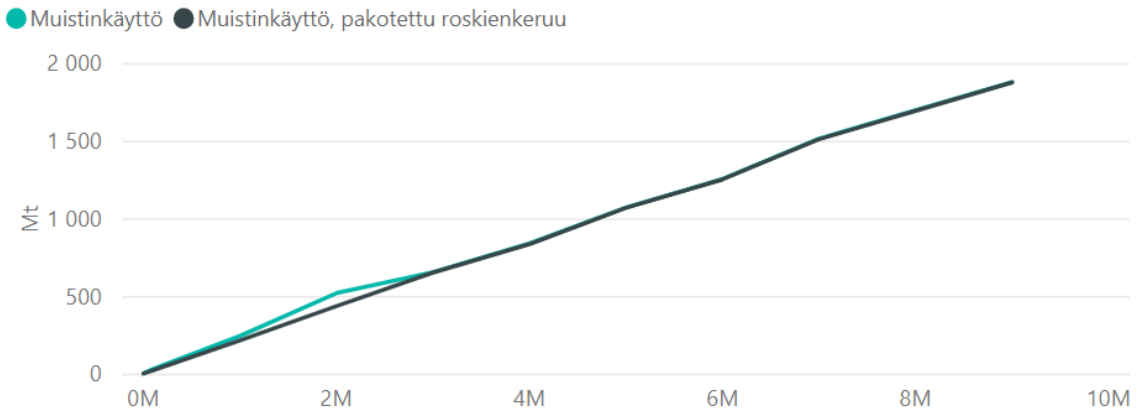
Kuvasta voidaan tehdä useita havaintoja. iPad Pron pieneen päivitykseen kuluva aika vaikuttaa kasvavan testin edetessä, joskin hieman ennen loppua aika laskee. MacBook Pron pieneen päivitykseen kuluttama aika käyttäytyy testin loppupuolella hieman

poukkoilevasti. TehoPC:n käyttämän aika pysyy melko vakaana testin alkupuolen, mutta lopussa hajonta alkaa kasvamaan.

Havainnot eivät suoraan kerro visualisointikomponentin vakaudesta tai käytöksestä, vaan sen varmistamiseksi tulee suorittaa lisätestejä pidempää testiaikaa käyttäen. Tähän ei kuitenkaan tämän diplomityön puitteissa ryhdytä.

6.9 Muistin kulutus

Testitapauksessa selvitettiin muistin kulutus eri datapistemäärillä. Alla olevassa kuvassa 18 on esitetty muistinkäyttö megatavuina datapistemäärän funktiona.



Kuva 18. *Muistinkäyttö megatavuina datapistemäärän funktiona*

Kuvasta nähdään, että muistin kulutus nousee lineaarisesti datapistemäärän kasvaessa.

6.10 Muistivuoto

Testitapauksessa pyrittiin havaitsemaan mahdollinen muistivuoto testipenkissä ja visualisointikomponentissa. Testitapaus suoritettiin aluksi aliluvun 4.9 testiaskelien mukaisesti, joskin testin kestoa rajoitti selaimen kehittäjätyökalujen suorituskykymittauksen maksimikesto. Testiajossa käytetyt lähtöarvot on esitelty taulukossa 37.

Taulukko 37 *Lyhyessä testissä käytetyt lähtöarvot*

Parametri	Arvo
Datapistemäärä täydessä päivityksessä	100 000
Datapistemäärä pienessä päivityksessä	10 000
Täyden päivityksen intervalli	20 000 ms
Pienen päivityksen intervalli	5 000 ms
Testin kesto	60 000 ms

Lyhyen testin aikana selaimen JavaScript-moottorin keon (engl. heap) koko kasvoi

hieman, noin puoli megatavua. Aikajanan ja pinosta otettujen kuvien (engl. snapshot) perusteella vaikuttaa siltä, että kasvu johtuu JavaScript-moottorin sisäisistä muistirakenteista, Three.js:n vasta tarvittaessa ladatuista moduuleista ja Reactin tavasta muistaa myös komponentin edellinen tila. Näiden havaintojen pohjalta vaikuttaisi siltä, että itse testipenkissä ja visualisointikomponentissa ei ole muistivuotoja.

Päätelmän ja visualisointikomponentille asetetun vakausvaatimuksen varmistamiseksi suoritettiin vielä pitkä kahdeksan tunnin testiajo, jonka aikana muistinkäyttö mitattiin aluksi ja lopuksi. Taulukossa 38 on esitelty pitkässä testissä käytetyt lähtöarvot ja taulukossa 39 muistin kulutus testin aluksi ja lopuksi.

Taulukko 38 *Pitkässä testissä käytetyt lähtöarvot*

Parametri	**Arvo**
Datapistemäärä täydessä päivityksessä	1 000 000
Datapistemäärä pienessä päivityksessä	10 000
Täyden päivityksen intervalli	60 000 ms
Pienen päivityksen intervalli	5 000 ms
Testin kesto	28 800 000 ms

Taulukko 39 *Muistin kulutus pitkän testin alussa ja lopussa*

Muistin kulutus aluksi	7,2 Mt
Muistin kulutus lopuksi	7,6 Mt

Taulukon 39 tuloksista nähdään selvästi, että muistin käyttö palaa testin lopuksi odotettuun suuruusluokkaan. Näin ollen voidaan pitää toteennäytettynä, että visualisointikomponentissa ei ole muistivuotoa.

7. JOHTOPÄÄTÖKSET

Tämän diplomityön tavoitteena oli toteuttaa WebGL:ää hyödyntävä prototyyppivisualisointikomponentti ja selvittää sen suorituskyky kattavasti. Komponentille ei asetettu varsinaisia suorituskykyvaatimuksia, mutta onnistumiselle suuntaa-antavana datapistemääränä pidettiin 100 000 datapistettä. Tarkasteltaessa suorituskykytestien tuloksia tällä datapistemäärällä voidaan visualisointikomponenttia pitää erittäin onnistuneena.

Kolmeen miljoonaan datapisteeseen saakka sekä tehokas pöytätietokone että Apple iPad Pro suoriutuivat mainiosti visualisoinnin käyttämisestä ja interaktioista, mutta visualisoidun datan päivittäminen hidastui kummallakin huomattavasti jo miljoonan datapisteen jälkeen. Noin kymmenen vuotta vanha Apple MacBook Pro pyöritti 100 000 datapisteen visualisointia ongelmitta, mutta miljoona pistettä muutti interaktiot jo hieman jähmeämmiksi, vaikkakin vielä käyttökelpoisia olivatkin.

Tässä diplomityössä suoritettiin testit vain kolmella erilaisella laitteella. Vaikka tulokset olivatkin kaikilla käytetyillä laitteilla lupaavia, kannattanee visualisointikomponenttia testata monipuolisemmin erilaisilla laitteilla ennen sen laajamittaista hyödyntämistä. Esimerkiksi matkapuhelimet ja Android-pohjaiset taulutietokoneet ovat laajalti käytössä olevia laitteita, joita ei tässä työssä käytetty lainkaan.

Visualisointikomponentin jatkokehityksen kannalta selvä parannuskohde on datan päivittämisen tehostaminen. Nykyinen ratkaisu luo kaikki datapisteitä vastaavat Three.js-objektit uusiksi, kun visualisoitava data muuttuu hiemankin. Yksi mahdollinen tapa parantaa suorituskykyä olisikin hyödyntää object pooling -mallia, jossa käytettyjä objekteja ei tuhota, vaan ne piilotetaan ruudulta ja käytetään myöhemmin uudelleen. Tällä vältetään turha objektien tuhoaminen ja luominen ja vältetään turha tiedonvaihto näytönohjaimen ja muun järjestelmän välillä.

Eräs mielenkiintoinen vertailu tulevaisuudessa toteutettavaksi on toteuttaa vastaava visualisointi ilman Three.js-kirjastoa ja verrata sen suorituskykyä tässä työssä saatuihin tuloksiin. Oletusarvoisesti kirjaston tarjoama yleishyödyllinen korkean tason rajapinta heikentää suorituskykyä.

Työn tulosten perusteella voidaan todeta, että WebGL:ää ja siten näytönohjainta hyödyntämällä voidaan luoda interaktiivisia visualisointeja merkittävän kokoisille datajoukoille ilman suorituskykyongelmia.

LÄHTEET

[1] Canvas API, Mozilla, Mar 18,2019, Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API, Viitattu: Mar 20,2019.

[2] K. Lukka, Kari Lukka: Konstruktiivinen tutkimusote, May 19,2014, Saatavissa: https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote/, Viitattu: Jul 1,2019.

[3] C. Ware, Information Visualization, 3rd ed. Boston: Morgan Kaufmann, 2012, 537 p.

[4] D.A. Keim, F. Mansmann, J. Schneidewind, H. Ziegler, Challenges in Visual Data Analysis, Tenth International Conference on Information Visualisation (IV'06), London, England, UK, 5-7 July 2006, IEEE, 5-7 July 20062006, pp. 9-16.

[5] V. Kuusela, A. Hyppönen, Tilastografiikan perusteet, Edita, Helsinki, 2000, 205 s.

[6] M. Friendly, D. Denis, The early origins and development of the scatterplot, Journal of the history of the behavioral sciences, Vol. 41, Iss. 2, 2005, pp. 103-130.

[7] J. S. Yi, Y. a. Kang, J. Stasko, Toward a Deeper Understanding of the Role of Interaction in Information Visualization, IEEE Transactions on Visualization and Computer Graphics, Vol. 13, Iss. 6, 2007, pp. 1224-1231.

[8] M. DiPierro, The Rise of JavaScript, Computing in Science & Engineering, Vol. 20, Iss. 1, 2018, pp. 9-10.

[9] J. Cuomo, JavaScript Everywhere and the Three Amigos (Into the wild BLUE yonder!), IBM, -08-142015, Saatavissa: https://www.ibm.com/developerworks/community/blogs/gcuomo/entry/javascript_everywhere_and_the_three_amigos?lang=en, Viitattu: May 6,2019.

[10] O. Campesato, SVG Pocket Primer, Mercury Learning & Information LLC, Bloomfield, 2016.

[11] SVG: Scalable Vector Graphics, Mozilla, Mar 18,2019, Saatavissa: https://developer.mozilla.org/en-US/docs/Web/SVG, Viitattu: Mar 20,2019.

[12] A. Bellamy-Royds, T. Bah, C. Lilley, D. Schulze & E. Willigers, , Scalable Vector Graphics (SVG) 2, W3C, May 1,2019, Saatavissa: https://svgwg.org/svg2-draft/Overview.html, Viitattu: May 6,2019.

[13] M.A. Perna, Canvas vs. SVG: Choosing the Right Tool for the Job, -03-23T10:00:19+00:002016, Saatavissa: https://www.sitepoint.com/canvas-vs-svg-choosing-the-right-tool-for-the-job/, Viitattu: Mar 20,2019.

[14] J. Marinacci, HTML Canvas Deep Dive, Saatavissa: https://joshondesign.com/p/books/canvasdeepdive/title.html, Viitattu: Mar 20,2019.

[15] WebGL: 2D and 3D graphics for the web, Mozilla, Mar 18,2019, Saatavissa: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API, Viitattu: Mar 20,2019.

[16] Khronos Releases Final WebGL 1.0 Specification, The Khronos Group, -03-03T07:00:00-08:002011, Saatavissa: https://www.khronos.org/news/press/khronos-releases-final-webgl-1.0-specification, Viitattu: Mar 20,2019.

[17] WebGL Stats, Saatavissa: https://webglstats.com, Viitattu: Mar 20,2019.

[18] M. Bostock, D3.js - Data-Driven Documents, 2019, Saatavissa: https://d3js.org/, Viitattu: Mar 24,2019.

[19] M. Bostock, V. Ogievetsky, J. Heer, D3: Data-Driven Documents, IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis), 2011, http://idl.cs.washington.edu/files/2011-D3-InfoVis.pdf.

[20] A Visualization Grammar, The Vega Project, Saatavissa: https://vega.github.io/vega/, Viitattu: March 24,2019.

[21] A. Satyanarayan, K. Wongsuphasawat, J. Heer, Declarative Interaction Design for Data Visualization, Proceedings of the 27th annual ACM symposium on user interface software and technology, Honolulu, Hawaii, USA, 5-8 October 2014, ACM, 5-8 October 20142014, pp. 669-678.

[22] Research Publications, The Vega Project, Saatavissa: https://vega.github.io/vega/about/research/, Viitattu: Mar 24,2019.

[23] R. Cabello, First public version. Still a lot to do, Apr 24,2010, Saatavissa: https://github.com/mrdoob/three.js/commit/a90c4e107ff6e3b148458c96965e876f9441b147, Viitattu: May 6,2019.

[24] R. Cabello, mrdoob/three.js: JavaScript 3D library, Apr 24,2010, Saatavissa: https://github.com/mrdoob/three.js/, Viitattu: May 6,2019.

[25] D. Bosnjak, What is three.js? -08-05T01:34:58.289Z2018, Saatavissa: https://medium.com/@pailhead011/what-is-three-js-7a03d84d9489, Viitattu: May 7,2019.

[26] P. Hunt, Why did we build React? – React Blog, Facebook, June 5,2013, Saatavissa: https://reactjs.org/blog/2013/06/05/why-react.html, Viitattu: May 7,2019.

[27] Facebook, Flux | Application Architecture for Building User Interfaces, Facebook, Saatavissa: https://facebook.github.io/flux/docs/in-depth-overview.html#content, .

[28] Create React App · Set up a modern web app by running one command. Facebook, 2019, Saatavissa: https://facebook.github.io/create-react-app/index.html, Viitattu: Mar 28,2019.

[29] D. Abramov, Redux · A Predictable State Container for JS Apps, Saatavissa: https://redux.js.org/, Viitattu: Jul 1,2019.

[30] 4 Types of Memory Leaks in JavaScript and How to Get Rid Of Them, Auth0, Saatavissa: https://auth0.com/blog/four-types-of-leaks-in-your-javascript-code-and-how-to-get-rid-of-them/, Viitattu: Apr 28,2019.

[31] Refs and the DOM – React, Facebook, Saatavissa: https://reactjs.org/docs/refs-and-the-dom.html, Viitattu: Apr 3,2019.

[32] three.js docs, Saatavissa: https://threejs.org/docs/, Viitattu: May 1,2019.

LIITE A: TESTITAPAUS: VISUALISOINNIN PIIRTOAIKA


```
{
    "name": "Measure render time",
    "id": "rendertime",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 1000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Render measurement start mark",
            "id": "render-start",
            "type": "hidden",
            "value": "render-meas-start"
        },
        {
            "name": "Render measurement end mark",
            "id": "render-end",
            "type": "hidden",
            "value": "render-meas-end"
        },
        {
            "name": "Render measurement",
            "id": "render-measurement",
            "type": "hidden",
            "value": "render-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
```

```
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Set re-render on data change",
            "resultAction": "ENABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Start render measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "render-start"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
        {
            "name": "End render measurement",
            "waitForEvent": "onReady",
            "testFunction": "createPerfMark",
            "parameters": [
                "render-end"
            ]
        },
        {
```

```
            "name": "Report render measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "render-start",
                "render-end",
                "render-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE B: TESTITAPAUS: DATAPISTEEN KOROSTAMINEN HIIREN LÄHEISYYDESTÄ

```
{
    "name": "Measure hover time",
    "id": "hovertime",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Hover X",
            "id": "hoverX",
            "type": "int",
            "default": 50,
            "min": 0,
            "max": 100
        },
        {
            "name": "Hover Y",
            "id": "hoverY",
            "type": "int",
            "default": 50,
            "min": 0,
            "max": 100
        },
        {
            "name": "Hover measurement start mark",
            "id": "hover-start",
            "type": "hidden",
            "value": "hover-meas-start"
        },
        {
            "name": "Hover measurement end mark",
            "id": "hover-end",
            "type": "hidden",
            "value": "hover-meas-end"
        },
        {
```

```
            "name": "Hover measurement",
            "id": "hover-measurement",
            "type": "hidden",
            "value": "hover-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
```

```
        {
            "name": "Wait for render to complete",
            "waitForEvent": "onReady"
        },
        {
            "name": "Start hover measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "hover-start"
            ]
        },
        {
            "name": "Dispatch synthetic mousemove event",
            "testFunction": "dispatchMouseMove",
            "parameters": [
                "hoverX",
                "hoverY"
            ]
        },
        {
            "name": "Wait for hover to complete",
            "waitForEvent": "onDataHover"
        },
        {
            "name": "End hover measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "hover-end"
            ]
        },
        {
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report hover measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "hover-start",
                "hover-end",
                "hover-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE C: TESTITAPAUS: DATAPISTEEN VALITSEMINEN KLIKKAAMALLA


```
{
    "name": "Measure click time",
    "id": "clicktime",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Click X",
            "id": "clickX",
            "type": "int",
            "default": 50,
            "min": 0,
            "max": 100
        },
        {
            "name": "Click Y",
            "id": "clickY",
            "type": "int",
            "default": 50,
            "min": 0,
            "max": 100
        },
        {
            "name": "Click measurement start mark",
            "id": "click-start",
            "type": "hidden",
            "value": "click-meas-start"
        },
        {
            "name": "Click measurement end mark",
            "id": "click-end",
            "type": "hidden",
            "value": "click-meas-end"
        },
        {
```

```
            "name": "Click measurement",
            "id": "click-measurement",
            "type": "hidden",
            "value": "click-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
```

```
        {
            "name": "Wait for render to complete",
            "waitForEvent": "onReady"
        },
        {
            "name": "Start click measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "click-start"
            ]
        },
        {
            "name": "Dispatch synthetic click event",
            "testFunction": "dispatchMouseClick",
            "parameters": [
                "clickX",
                "clickY"
            ]
        },
        {
            "name": "Wait for click to complete",
            "waitForEvent": "onDataClick"
        },
        {
            "name": "End click measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "click-end"
            ]
        },
        {
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report click measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "click-start",
                "click-end",
                "click-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE D: TESTITAPAUS: VISUALISOINNIN PANOROINTI RAAHAAMALLA


```
{
    "name": "Measure drag fps",
    "id": "dragfps",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Drag start X",
            "id": "startX",
            "type": "hidden",
            "default": 0,
            "min": 0,
            "max": 100
        },
        {
            "name": "Drag start Y",
            "id": "startY",
            "type": "hidden",
            "default": 0,
            "min": 0,
            "max": 100
        },
        {
            "name": "Drag end X",
            "id": "endX",
            "type": "hidden",
            "default": 100,
            "min": 0,
            "max": 100
        },
        {
            "name": "Drag end Y",
            "id": "endY",
            "type": "hidden",
            "default": 100,
```

```
            "min": 0,
            "max": 100
        },
        {
            "name": "Drag steps",
            "id": "steps",
            "type": "hidden",
            "default": 240,
            "min": 10,
            "max": 1200
        },
        {
            "name": "Drag measurement start mark",
            "id": "drag-start",
            "type": "hidden",
            "value": "drag-meas-start"
        },
        {
            "name": "Drag measurement end mark",
            "id": "drag-end",
            "type": "hidden",
            "value": "drag-meas-end"
        },
        {
            "name": "Drag measurement",
            "id": "drag-measurement",
            "type": "hidden",
            "value": "drag-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
```

```
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
        {
            "name": "Wait for render to complete",
            "waitForEvent": "onReady"
        },
        {
            "name": "Start drag time measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "drag-start"
            ]
        },
        {
            "name": "Dispatch synthetic mousedown event",
            "testFunction": "dispatchMouseDown"
        },
        {
            "name": "Simulate drag",
            "testFunction": "simulateMouseMove",
            "parameters": [
                "startX",
                "startY",
                "endX",
                "endY",
                "steps"
            ]
        },
        {
            "name": "Dispatch synthetic mouseup event",
            "testFunction": "dispatchMouseUp"
        },
        {
            "name": "Wait for drag to complete",
            "waitForEvent": "onDragEnd"
        },
        {
            "name": "End hover measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "drag-end"
            ]
        },
        {
```

```
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report drag measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "drag-start",
                "drag-end",
                "drag-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report drag framecount",
            "testFunction": "countFramesBetweenMarks",
            "parameters": [
                "drag-start",
                "drag-end"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE E: TESTITAPAUS: VISUALISOINNIN ZOOMAUS HIIREN RULLALLA


```
{
    "name": "Measure wheelzoom time",
    "id": "wheelzoomtime",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 50000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Zoom steps",
            "id": "wheel-steps",
            "type": "int",
            "default": 20,
            "min": -40,
            "max": 40
        },
        {
            "name": "Wheel zoom measurement start mark",
            "id": "wheelzoom-start",
            "type": "hidden",
            "value": "wheelzoom-meas-start"
        },
        {
            "name": "Wheel zoom measurement end mark",
            "id": "wheelzoom-end",
            "type": "hidden",
            "value": "wheelzoom-meas-end"
        },
        {
            "name": "Wheel zoom measurement",
            "id": "wheelzoom-measurement",
            "type": "hidden",
            "value": "wheelzoom-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
```

```
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
        {
            "name": "Wait for render to complete",
            "waitForEvent": "onReady"
        },
        {
            "name": "Start wheelzoom measurement",
            "testFunction": "createPerfMark",
            "parameters": [
```

```
                "wheelzoom-start"
            ]
        },
        {
            "name": "Simulate zoom",
            "testFunction": "wheelZoomMultiple",
            "parameters": [
                "wheel-steps"
            ]
        },
        {
            "name": "End wheelzoom measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "wheelzoom-end"
            ]
        },
        {
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report wheelzoom measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "wheelzoom-start",
                "wheelzoom-end",
                "wheelzoom-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report zoom framecount",
            "testFunction": "countFramesBetweenMarks",
            "parameters": [
                "wheelzoom-start",
                "wheelzoom-end"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE F: TESTITAPAUS: DATAN PÄIVITTÄMINEN


```
{
    "name": "Measure data update time",
    "id": "datachangetime",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Render measurement start mark",
            "id": "update-start",
            "type": "hidden",
            "value": "update-meas-start"
        },
        {
            "name": "Render measurement end mark",
            "id": "update-end",
            "type": "hidden",
            "value": "update-meas-end"
        },
        {
            "name": "Render measurement",
            "id": "update-measurement",
            "type": "hidden",
            "value": "update-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
```

```
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Start data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-start"
            ]
        },
        {
            "name": "Generate new data",
            "testFunction": "createTestData",
            "parameters": [
                "itemCount",
                "xdev",
                "ydev"
            ],
            "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
        },
        {
            "name": "End data generation measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "generation-end"
            ]
        },
        {
            "name": "Start render measurement",
            "testFunction": "createPerfMark",
            "parameters": [
                "update-start"
            ]
        },
        {
            "name": "Dispatch test data",
            "testFunction": "passthroughOne",
            "parameters": [
                "TEST_DATA"
            ],
            "resultAction": "SET_TEST_DATA",
            "asyncDispatch": true
        },
        {
            "name": "End render measurement",
            "waitForEvent": "onReady",
            "testFunction": "createPerfMark",
            "parameters": [
                "update-end"
            ]
        },
        {
            "name": "Report render measurement result",
            "testFunction": "measureMarks",
```

```
            "parameters": [
                "update-start",
                "update-end",
                "update-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        },
        {
            "name": "Report generation measurement result",
            "testFunction": "measureMarks",
            "parameters": [
                "generation-start",
                "generation-end",
                "generation-measurement"
            ],
            "resultAction": "PUBLISH_TEST_RESULT"
        }
    ]
}
```

LIITE G: TESTITAPAUS: JATKUVA DATAN SYÖTTÄMINEN


```
{
    "name": "Continous data update",
    "id": "continousdata",
    "description": "Not used",
    "testModule": null,
    "autoUpdate": true,
    "parameters": [
        {
            "name": "Items",
            "id": "itemCount",
            "type": "int",
            "default": 1000000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "Increment items",
            "id": "incrementItemCount",
            "type": "int",
            "default": 10000,
            "min": 0,
            "max": 10000000
        },
        {
            "name": "X Deviation",
            "id": "xdev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Y Deviation",
            "id": "ydev",
            "type": "int",
            "default": 100000,
            "min": 0,
            "max": 200000
        },
        {
            "name": "Full update interval",
            "id": "full-interval",
            "type": "int",
            "default": 60000,
            "min": 1,
            "max": 1000000
        },
        {
            "name": "Incremental update interval",
            "id": "incr-interval",
            "type": "int",
            "default": 5000,
            "min": 1,
            "max": 1000000
        },
        {
            "name": "Test duration",
            "id": "duration",
            "type": "int",
            "default": 28800000,
```

```
            "min": 1,
            "max": 100000000
        },
        {
            "name": "Render measurement start mark",
            "id": "update-start",
            "type": "hidden",
            "value": "update-meas-start"
        },
        {
            "name": "Render measurement end mark",
            "id": "update-end",
            "type": "hidden",
            "value": "update-meas-end"
        },
        {
            "name": "Render measurement",
            "id": "update-measurement",
            "type": "hidden",
            "value": "update-time"
        },
        {
            "name": "Increment render measurement start mark",
            "id": "increment-update-start",
            "type": "hidden",
            "value": "incr-update-meas-start"
        },
        {
            "name": "Increment render measurement end mark",
            "id": "increment-update-end",
            "type": "hidden",
            "value": "incr-update-meas-end"
        },
        {
            "name": "Increment render measurement",
            "id": "increment-update-measurement",
            "type": "hidden",
            "value": "incr-update-time"
        },
        {
            "name": "Data generation measurement start mark",
            "id": "generation-start",
            "type": "hidden",
            "value": "gen-meas-start"
        },
        {
            "name": "Data generation measurement end mark",
            "id": "generation-end",
            "type": "hidden",
            "value": "gen-meas-end"
        },
        {
            "name": "Data generation measurement",
            "id": "generation-measurement",
            "type": "hidden",
            "value": "data-generation-time"
        },
        {
            "name": "Increment data generation measurement start mark",
            "id": "increment-generation-start",
            "type": "hidden",
            "value": "incr-gen-meas-start"
        },
        {
            "name": "Increment data generation measurement end mark",
            "id": "increment-generation-end",
```

```
            "type": "hidden",
            "value": "incr-gen-meas-end"
        },
        {
            "name": "Increment data generation measurement",
            "id": "increment-generation-measurement",
            "type": "hidden",
            "value": "incr-data-generation-time"
        }
    ],
    "steps": [
        {
            "name": "Clear performance measurements",
            "testFunction": "clearPerf"
        },
        {
            "name": "Disable re-render on data change",
            "resultAction": "DISABLE_RERENDER_ON_DATA_CHANGE"
        },
        {
            "name": "Run once per minute",
            "type": "stepGroup",
            "initialDelay": 0,
            "intervalParameter": "full-interval",
            "durationParameter": "duration",
            "steps": [
                {
                    "name": "Clear per minute performance measurements",
                    "testFunction": "clearPerfWithName",
                    "parameters": [
                        "update-start",
                        "update-end",
                        "update-measurement",
                        "generation-start",
                        "generation-end",
                        "generation-measurement"
                    ]
                },
                {
                    "name": "Start data generation measurement",
                    "testFunction": "createPerfMark",
                    "parameters": [
                        "generation-start"
                    ]
                },
                {
                    "name": "Generate new data",
                    "testFunction": "createTestData",
                    "parameters": [
                        "itemCount",
                        "xdev",
                        "ydev"
                    ],
                    "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
                },
                {
                    "name": "End data generation measurement",
                    "testFunction": "createPerfMark",
                    "parameters": [
                        "generation-end"
                    ]
                },
                {
                    "name": "Start render measurement",
                    "testFunction": "createPerfMark",
                    "parameters": [
```

```
                    "update-start"
                ]
            },
            {
                "name": "Dispatch test data",
                "testFunction": "passthroughOne",
                "parameters": [
                    "TEST_DATA"
                ],
                "resultAction": "SET_TEST_DATA",
                "asyncDispatch": true
            },
            {
                "name": "End render measurement",
                "waitForEvent": "onReady",
                "testFunction": "createPerfMark",
                "parameters": [
                    "update-end"
                ]
            },
            {
                "name": "Report render measurement result",
                "testFunction": "measureMarks",
                "parameters": [
                    "update-start",
                    "update-end",
                    "update-measurement"
                ],
                "resultAction": "PUBLISH_TEST_RESULT"
            },
            {
                "name": "Report generation measurement result",
                "testFunction": "measureMarks",
                "parameters": [
                    "generation-start",
                    "generation-end",
                    "generation-measurement"
                ],
                "resultAction": "PUBLISH_TEST_RESULT"
            }
        ]
    },
    {
        "name": "Run once per 5 seconds",
        "type": "stepGroup",
        "initialDelay": 5000,
        "intervalParameter": "incr-interval",
        "durationParameter": "duration",
        "steps": [
            {
                "name": "Clear per 5s performance measurements",
                "testFunction": "clearPerfWithName",
                "parameters": [
                    "increment-update-start",
                    "increment-update-end",
                    "increment-update-measurement",
                    "increment-generation-start",
                    "increment-generation-end",
                    "increment-generation-measurement"
                ]
            },
            {
                "name": "Start increment data generation measurement",
                "testFunction": "createPerfMark",
                "parameters": [
                    "increment-generation-start"
```

```
                ]
            },
            {
                "name": "Generate new increment data",
                "testFunction": "createIncrementData",
                "parameters": [
                    "incrementItemCount",
                    "xdev",
                    "ydev"
                ],
                "resultAction": "TEST_ACTION//SAVE_RUNTIME_DATA//TEST_DATA"
            },
            {
                "name": "End increment data generation measurement",
                "testFunction": "createPerfMark",
                "parameters": [
                    "increment-generation-end"
                ]
            },
            {
                "name": "Start increment render measurement",
                "testFunction": "createPerfMark",
                "parameters": [
                    "increment-update-start"
                ]
            },
            {
                "name": "Dispatch test data",
                "testFunction": "passthroughOne",
                "parameters": [
                    "TEST_DATA"
                ],
                "resultAction": "SET_TEST_DATA",
                "asyncDispatch": true
            },
            {
                "name": "End increment render measurement",
                "waitForEvent": "onReady",
                "testFunction": "createPerfMark",
                "parameters": [
                    "increment-update-end"
                ]
            },
            {
                "name": "Report render measurement result",
                "testFunction": "measureMarks",
                "parameters": [
                    "increment-update-start",
                    "increment-update-end",
                    "increment-update-measurement"
                ],
                "resultAction": "PUBLISH_TEST_RESULT"
            },
            {
                "name": "Report generation measurement result",
                "testFunction": "measureMarks",
                "parameters": [
                    "increment-generation-start",
                    "increment-generation-end",
                    "increment-generation-measurement"
                ],
                "resultAction": "PUBLISH_TEST_RESULT"
            }
        ]
    }
]
```

```
}
```